Transformations éditoriales avancées

LHQ1C1M1
Master 2 Humanités numériques

XSLT

eXstensible Stylesheet Language Transformations

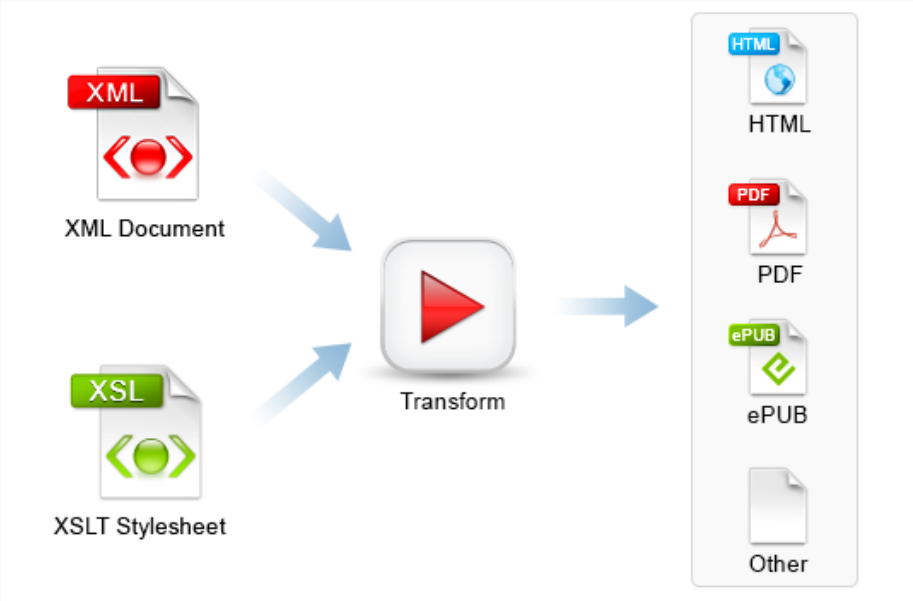

Figure 1 : transformation XSLT

Une première transformation

Avec XSLT, on peut très facilement transformer ceci :

```
<head rend="alignCenter">Chapitre 1 : <emph>XSLT</emph></head>
```

et produire :

```
<h1 class="alignCenter">Chapitre 1 : <em>XSLT</em></h1>
```

Pour cela il faut simplement :

Une première transformation

Avec XSLT, on peut très facilement transformer ceci :

```
<head rend="alignCenter">Chapitre 1 : <emph>XSLT</emph></head>
```

et produire :

```
<h1 class="alignCenter">Chapitre 1 : <em>XSLT</em></h1>
```

Pour cela il faut simplement :

- transformer l'élément TEI <head/> en un élément (x)HTML <h1/>;

Une première transformation

Avec XSLT, on peut très facilement transformer ceci :

```
<head rend="alignCenter">Chapitre 1 : <emph>XSLT</emph></head>
```

et produire :

```
<h1 class="alignCenter">Chapitre 1 : <em>XSLT</em></h1>
```

Pour cela il faut simplement :

- transformer l'élément TEI `<head/>` en un élément (x)HTML `<h1/>` ;
- transformerl'attribut `@class` en un attribut `@rend`.

Avec XSLT

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xpath-default-namespace="http://www.tei-c.org/ns/1.0"
    version="2.0">

  <xsl:template match="head">
    <h1 class="{ @rend }">
      <xsl:apply-templates/>
    </h1>
  </xsl:template>

</xsl:stylesheet>
```

Anatomie d’une feuille de transformation

- Une feuille de style XSLT est un document XML
- l’élément racine est au choix `<stylesheet/>` ou `<transform/>`
- les éléments XSLT sont dans l’espace de nom `http://www.w3.org/1999/XSL/Transform`
- Cet espace de noms est généralement associé au préfixe `xsl`
- la version de la norme XSLT utilisée est indiquée sur l’élément racine avec l’attribut `@version` (`1.0`, `2.0` ou `3.0`)

Une feuille de transformation minimaliste

Exercice

Appliquer la feuille XSLT suivante sur un document XML. Que remarque-t-on ?

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

</xsl:stylesheet>
```

Feuille de transformation minimaliste

- le résultat n’est pas un document XML
- les balises et les attributs ont été supprimés

Processeur XSLT

Le processeur XSLT traite le document XML à transformer de la manière suivante :

- il lit les nœuds dans l'ordre, en commençant par l'élément racine
- pour chaque nœud, il vérifie si une règle modèle (*template*) existe
  - si aucune ne correspond, il traite les nœuds fils.
  - si aucun élément ne reste à traiter, il retourne le contenu textuel
  - si une règle existe, elle est appliquée
- l'ordre de déclaration des *templates* n'a pas d'importance

Règles modèles `<xsl:template/>`

Une feuille de transformation XSLT repose sur l'utilisation de **règles modèles** (`<xsl:template/>`). Toute règle modèle est composée d'un **motif** (`@match`) qui identifie les nœuds à transformer et d'un **modèle** qui est appliqué lorsque le motif est trouvé dans l'arbre.

Le **motif** le plus simple est le nom d'un élément :

```
<xsl:template match="head"><!-- @match : expression XPath  -->
    <!-- modèle : le contenu de xsl:template -->
    Un titre
</xsl:template>
```

Règles modèles <xsl:template/>

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xpath-default-namespace="http://www.tei-c.org/ns/1.0"
    version="2.0">
    <!-- règle modèle -->
    <xsl:template match="head">
        Un titre
    </xsl:template>
</xsl:stylesheet>
```

Ce qui donne en sortie (*literal data characters*) :

```
<?xml version="1.0"?>
Un titre
```

Règles modèles <xsl:template/>

Bien évidemment, on peut aussi imprimer des nœuds en sortie (*literal result elements*)

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xpath-default-namespace="http://www.tei-c.org/ns/1.0"
    version="2.0">
    <xsl:template match="head">
        <h1>un titre</h1>
    </xsl:template>
</xsl:stylesheet>
```

Le résultat en sortie sera alors :

```
<h1>un titre</h1>
```

Déclarer un élément `<xsl:element/>`

On peut déclarer un élément de différentes manières :

```
<xsl:template match="head">
    <h1>Un titre</h1>
</xsl:template>
```

Il est possible de faire la même chose avec la balise `<xsl:element/>`

```
<xsl:template match="head">
    <xsl:element name="h1">Un titre</xsl:element>
</xsl:template>
```

Déclarer un attribut <xsl:attribute/>

Tout comme les éléments, plusieurs méthodes existent pour déclarer des attributs :

- soit en l'ajoutant directement :

```
<xsl:template match="head">
    <h1 class="center">Un titre</h1>
</xsl:template>
```

- soit avec la balise <xsl:attribute/>

```
<xsl:template match="head">
    <xsl:element name="h1">
        <xsl:attribute name="class">center</xsl:attribute>
        Un titre
    </xsl:element>
</xsl:template>
```

Déclarer un attribut <xsl:attribute/>

On peut également définir la valeur de l'attribut avec une expression XPath :

```
<xsl:template match="head">
    <h1 class="{ ./@rend }">
        Un titre
    </h1>
</xsl:template>
```

ou

```
<xsl:template match="head">
    <xsl:element name="h1">
        <xsl:attribute name="class" select="@rend"/>
        Un titre
    </xsl:element>
</xsl:template>
```

Appliquer des modèles `<xsl:apply-templates/>`

Nous l'avons vu, le processeur XSLT parcours le document XML de haut en bas. Les règles modèles sont donc appliquées dans l'ordre dans lequel apparaissent les éléments dans l'arbre. Cela signifie qu'une règle modèle d'un élément parent est appliquée avant les règles modèles des nœuds fils.

Les règles modèles peuvent de ce fait empêcher le traitement d'éléments particuliers. Dans les exemples précédents, elles empêchent implicitement le traitement des nœuds contenus dans l'élément `<head/>`.

Appliquer des modèles <xsl:apply-templates/>

```
<head rend="alignCenter">Chapitre 1 : <emph>XSLT</emph></head>

<xsl:template match="head">
    <h1 class="{ ./@rend }">Un titre</h1>
</xsl:template>
```

Le contenu de l'élément <head/> est remplacé par "Un titre" :

- le texte "Chapitre 1 : " est ignoré
- de même l'élément <emph/> et son contenu

Implicitement, on a demandé au processeur XSLT de ne pas descendre plus loin dans l'arborescence du nœud <head/> en donnant l'instruction d'imprimer "Un titre" à l'intérieur d'un élément <h1/>

Appliquer des modèles `<xsl:apply-templates/>`

La balise `<xsl:apply-templates/>` signifie « applique les autres règles modèles au contenu de l'élément courant ».

S'il n'y a pas de règle modèle dont le **motif** correspond au nœud, une règle modèle intégrée est utilisée (retourne la valeur textuelle.)

Appliquer des modèles <xsl:apply-templates/>

```
<xsl:template match="head">
    <h1 class="{ ./@rend }"><xsl:apply-templates/></h1>
</xsl:template>

<xsl:template match="emph">
    <em><xsl:apply-templates/></em>
</xsl:template>
```

résultat :

```
<h1 class="alignCenter">Chapitre 1 : <em>XSLT</em></h1>
```

Appliquer des modèles <xsl:apply-templates/>

Par défaut, l'instruction `<xsl:apply-templates/>` examine, dans l'ordre, les enfants du nœud contextuel. Cependant, il est possible de modifier ce comportement avec l'attribut `@select`. Il permet d'indiquer au processeur quel(s) autre(s) nœud(s) doi(ven)t être traité(s).

```
<xsl:template match="head">
    <xsl:apply-templates select="emph"/>
</xsl:template>

<xsl:template match="emph">
    <em><xsl:apply-templates/></em>
</xsl:template>
```

résultat :

```
<em>XSLT</em>
```

Valeur textuelle d'un nœud <xsl:value-of/>

Lorsque l'on souhaite uniquement accéder à la valeur textuelle d'un nœud, on peut recourir à l'élément <xf:value-of />. La valeur textuelle d'un nœud correspond au contenu d'un élément une fois que toutes les balises ont été retirées.

```
<xsl:template match="head">
    <xsl:value-of select="."/>
</xsl:template>
```

résultat :

```
Chapitre 1 : XSLT
```

Copier <xsl:copy-of/> et <xsl:copy-of/>

L'élément <xsl:copy-of/> permet d'ajouter dans l'arbre de sortie un élément de l'arbre d'entrée, sans le modifier.

```
<xsl:template match="head">
    <h1 class="{ ./@rend }"><xsl:apply-templates/></h1>
    <xsl:copy-of select="."/>
</xsl:template>

<xsl:template match="emph">
    <em><xsl:apply-templates/></em>
</xsl:template>
```

Résultat :

```
<h1 class="alignCenter">Chapitre 1 : <em>XSLT</em></h1>
<head rend="alignCenter">Chapitre 1 : <emph>XSLT</emph></head>
```

Copier <xsl:copy-of/> et <xsl:copy-of/>

L'élément <xsl:copy/> crée une paire de balises (ouvrante et fermante) dont le nom est celui de l'élément courant, mais ni les nœuds descendants, ni les attributs ne sont copiés.

```
<xsl:template match="head">
    <xsl:copy><xsl:apply-templates/></xsl:copy>
</xsl:template>

<xsl:template match="emph">
    <em><xsl:apply-templates/></em>
</xsl:template>
```

Résultat :

```
<head>Chapitre 1 : <em>XSLT</em></head>
```

Règles intégrées

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   exclude-result-prefixes="xs"
   version="2.0">
   <xsl:template match="text() | @*">
      <xsl:value-of select="."/>
   </xsl:template>
   <xsl:template match="* | /">
      <xsl:apply-templates/>
   </xsl:template>
   <xsl:template match="processing-instruction()|comment()"/>
</xsl:stylesheet>
```