Transformations éditoriales avancées

LHQ1C1M1
Master 2 Humanités numériques

XSLT

eXstensible Stylesheet Language Transformations

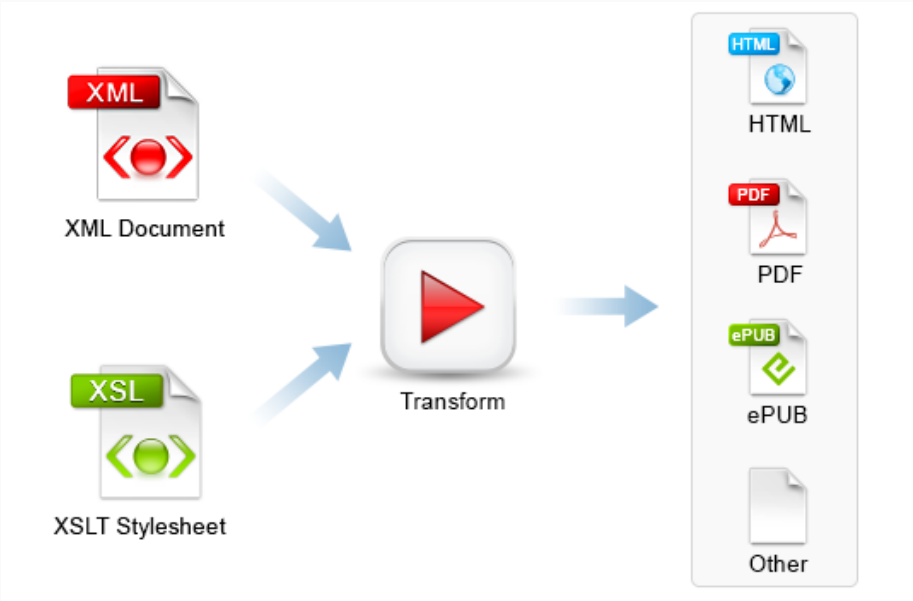

Figure 1 : transformation XSLT

Une première transformation

Avec XSLT, on peut très facilement transformer ceci :

```
<ref target="www.w3.org/TR/xslt-30/">The <emph>XSLT</emph> Standa
```

et produire :

```
<a href="www.w3.org/TR/xslt-30/">The <em>XSLT</em> Standard</a>
```

Pour cela il faut simplement :

Une première transformation

Avec XSLT, on peut très facilement transformer ceci :

```
<ref target="www.w3.org/TR/xslt-30/">The <emph>XSLT</emph> Standa
```

et produire :

```
<a href="www.w3.org/TR/xslt-30/">The <em>XSLT</em> Standard</a>
```

Pour cela il faut simplement :

- transformer l'élément TEI `<ref/>` en un élément (X)HTML `<a/>`

Une première transformation

Avec XSLT, on peut très facilement transformer ceci :

```
<ref target="www.w3.org/TR/xslt-30/">The <emph>XSLT</emph> Standa
```

et produire :

```
<a href="www.w3.org/TR/xslt-30/">The <em>XSLT</em> Standard</a>
```

Pour cela il faut simplement :

- transformer l'élément TEI `<ref/>` en un élément (X)HTML `<a/>`
- transformer l'attribut `@target` en un attribut `@href`.

Une première transformation

Avec XSLT, on peut très facilement transformer ceci :

```
<ref target="www.w3.org/TR/xslt-30/">The <emph>XSLT</emph> Standa
```

et produire :

```
<a href="www.w3.org/TR/xslt-30/">The <em>XSLT</em> Standard</a>
```

Pour cela il faut simplement :

- transformer l'élément TEI `<ref/>` en un élément (X)HTML `<a/>`
- transformer l'attribut `@target` en un attribut `@href`.
- transformer l'élément TEI `<emph/>` en un élément (X)HTML `<em/>`

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="ref">
    <a href="{ @target }"><xsl:apply-templates/></a>
  </xsl:template>

  <xsl:template match="emph">
    <em><xsl:apply-templates/></em>
  </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 `<xsl:stylesheet/>` ou `<xs:transform/>`
- les éléments XSLT sont dans l’espace de nom `http://www.w3.org/1999/XSL/Transform`
- Cet espace de nom 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`)

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="ref"><!-- @match : expression XPath  -->
    <!-- modèle : le contenu de xsl:template -->
    <a href="{ @target }"><xsl:apply-templates/></a>
</xsl:template>
```

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(x) nœud(s), une règle modèle intégrée est utilisée : elle retourne la valeur textuelle du nœud.

Appliquer des modèles - les règles intégrées

Appliquez la feuille XSLT suivante sur un document XML.

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

</xsl:stylesheet>
```

Appliquer des modèles - les règles intégrées

Que remarque-t-on ?

Appliquer des modèles - les règles intégrées

Que remarque-t-on ?

- le résultat n’est pas un document XML

Appliquer des modèles - les règles intégrées

Que remarque-t-on ?

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

Appliquer des modèles - les 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>
```

Appliquer des modèles

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="ref">
    <xsl:apply-templates select="emph"/>
</xsl:template>

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

résultat : `<em>XSLT</em>`

Appliquer du texte - *literal data characters*

Aucune obligation cependant d'appliquer des *templates* :

```
<xsl:template match="ref">
    XML
</xsl:template>
```

Le contenu de l'élément `<ref/>` est remplacé par `"XML"` :

- le texte et l'élément `<emph/>` sont ignorés

Implicitement, on a demandé au processeur XSLT de ne pas descendre plus loin dans l'arborescence du nœud `<ref/>` en donnant l'instruction d'imprimer `"XML"`.

Déclarer un élément

Bien évidemment, on peut aussi ajouter 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="ref">
        <a>XML</a>
    </xsl:template>
</xsl:stylesheet>
```

Le résultat en sortie sera alors : `<a>XML</a>`

Déclarer un élément <xsl:element/>

On peut déclarer un élément de différentes manières :

```
<xsl:template match="ref">
    <a><xsl:apply-templates/></a>
</xsl:template>

<xsl:template match="ref">
    <xsl:element name="a"><xsl:apply-templates/></xsl:element>
</xsl:template>
```

Déclarer un attribut <xsl:attribute/>

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

```
<xsl:template match="ref">
  <a href="http://unlien.org">Un lien</a>
</xsl:template>

<xsl:template match="ref">
  <xsl:element name="a">
    <xsl:attribute name="href">http://unlien.org</xsl:attribute>
    <xsl:apply-templates/>
  </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="ref">
    <a href="{ ./@target }">
        <xsl:apply-templates/>
    </a>
</xsl:template>

<xsl:template match="ref">
    <xsl:element name="a">
        <xsl:attribute name="href" select="@target"/>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

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

Lorsque l'on souhaite uniquement accéder à la valeur textuelle d'un nœud, on utilise l'élément <xsl: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="ref">
    <xsl:value-of select="."/>
</xsl:template>
```

résultat : The XSLT Standard