

Formation Edition numérique

La publication web

Simon Gabay



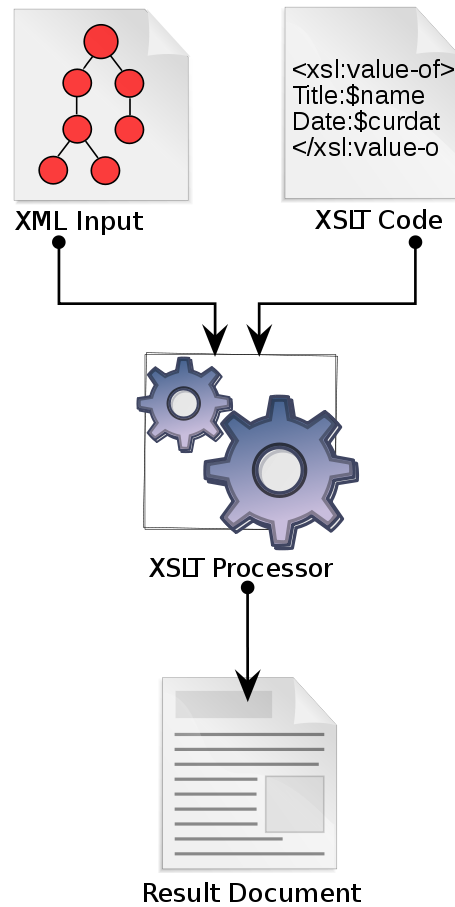
XSLT

XSLT

XSLT (*eXtensible Stylesheet Language Transformations*) est un langage de transformation XML de type fonctionnel. Il permet de modifier un document XML ou de le transformer en un autre document (CSV, HTML, LaTeX...).

Plus d'informations sur [Wikipedia](#)

La transformation



Source: [Wikipedia](#)

Une feuille de style XSLT minimale

```
<?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:stylesheet>
```

Deux remarques:

1. C'est un langage qui s'écrit en XML (on reconnaît les chevrons, les éléments, les attributs...)
2. Il existe plusieurs versions (1, 2 et 3) qui sont légèrement différentes. Il faut donc choisir le bon processeur XSLT (cf. image *supra* pour la bonne version).

XML et espace de nom

Dans un document XML-TEI, nous avons vu qu'il y a le langage d'une part (XML) et le vocabulaire d'une autre (TEI). Il faudrait donc préciser que les éléments XML utilisés relèvent de cet *espace de nom* à chaque fois au moyen d'un préfixe

```
<tei:TEI xmlns:tei="http://www.tei-c.org/ns/1.0">  
  <tei:teiHeader>  
    <tei:fileDesc>  
      <tei:titleStmt>  
        <tei:title>...
```

Il est possible de faire l'économie de cette précision en déclarant une seule fois l'espace de nom sur l'élément racine:

```
<TEI xmlns="http://www.tei-c.org/ns/1.0">  
  <teiHeader>  
    <fileDesc>  
      <titleStmt>  
        <title>...
```

XSLT et espace de nom

```
<?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:stylesheet>
```

Comme sur le document XML, il faut donc préciser l'espace de nom par défaut pour la feuille XSL, car il y en a un autre: `xsl:` .

```
<?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"
  xpath-default-namespace="http://www.tei-c.org/ns/1.0">

</xsl:stylesheet>
```

XSLT: principe de base

Une feuille XSLT ne modifie pas un fichier: il retire toutes les balises. Si l'on veut garder ou modifier un élément, il faut ajouter une règle. Pour ce faire, on utilise l'élément `xsl:template` :

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

Pour "copier-coller" l'élément `<head>` de cet exemple

```
<head>Sonetz</head>
```

il faut le remettre:

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


Une première modification

Pour changer l'élément `<head>` en élément `<h1>` :

```
<head>Sonetz</head>
```

il faut appliquer la règle suivante:

```
<xsl:template match="head">  
  <h1><xsl:apply-templates/></h1>  
</xsl:template>
```

On traduit ainsi le document XML en HTML:

```
<h1>Sonetz</h1>
```

Enchaîner les règles

Il faut donc appliquer une règle par élément que l'on veut modifier:

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

<xsl:template match="head">
  <h1>
    <xsl:apply-templates/>
  </h1>
</xsl:template>

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

Ordre

L'ordre de la feuille de style importe peu: ce qui compte est l'ordre du document XML.

```
<div>
  <head>Titre</head>
  <p>Un paragraphe</p>
  <p>Un autre paragraphe</p>
</div>
```

Ce document XML peut être traité par la série de règles suivante:

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

<xsl:template match="head">
  <h1><xsl:apply-templates/></h1>
</xsl:template>

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

Une page HTML incomplète

Rappelons qu'une page HTML minimale est la suivante:

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
  </head>
  <body>mon texte</body>
</html>
```

Notre résultat obtenu ne suffit pas: nous n'avons construit que l'intérieur du `<body>` :

```
<div>
  <head>Titre</head>
  <p>Un paragraphe</p>
  <p>Un autre paragraphe</p>
</div>
```

Une page HTML complète

Ces informations doivent entourer l'élément racine, que l'on désigne de manière conventionnelle avec un slash (/):

```
<xsl:template match="/">
  <html>
    <head>
      <title>Exercice XSLT</title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
```

Une transformation XML-TEI -> HTML minimale

Pour modifier un document XML-TEI avec uniquement une `<div>` :

```
<?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"
  xpath-default-namespace="http://www.tei-c.org/ns/1.0">

  <xsl:template match="/">
    <html>
      <head>
        <title>Exercice XSLT</title>
      </head>
      <body><xsl:apply-templates/></body>
    </html>
  </xsl:template>

  <xsl:template match="div">
    <div><xsl:apply-templates/></div>
  </xsl:template>

</xsl:stylesheet>
```

XPath

Précision

Dans la règle que nous avons mentionné au tout début, il convient de corriger une approximation

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

Dans cet exemple `unElement` n'est pas exactement le nom d'un élément, mais une fonction XPath.

XPath

XPath est un langage de requête pour localiser une portion d'un document XML.

Plus d'informations sur [Wikipedia](#)

Principe de base

Dans le document suivant, chaque élément, chaque élément est accessible par un chemin (en anglais *path*, d'où *XPath*):

```
<a>
  <b>
    <c>bla bla bla</c>
    <c>ta ta ta</c>
  </b>
  <d>gna gna gna</d>
</a>
```

Pour `<d>`, le chemin est le suivant:

```
/a/d
```

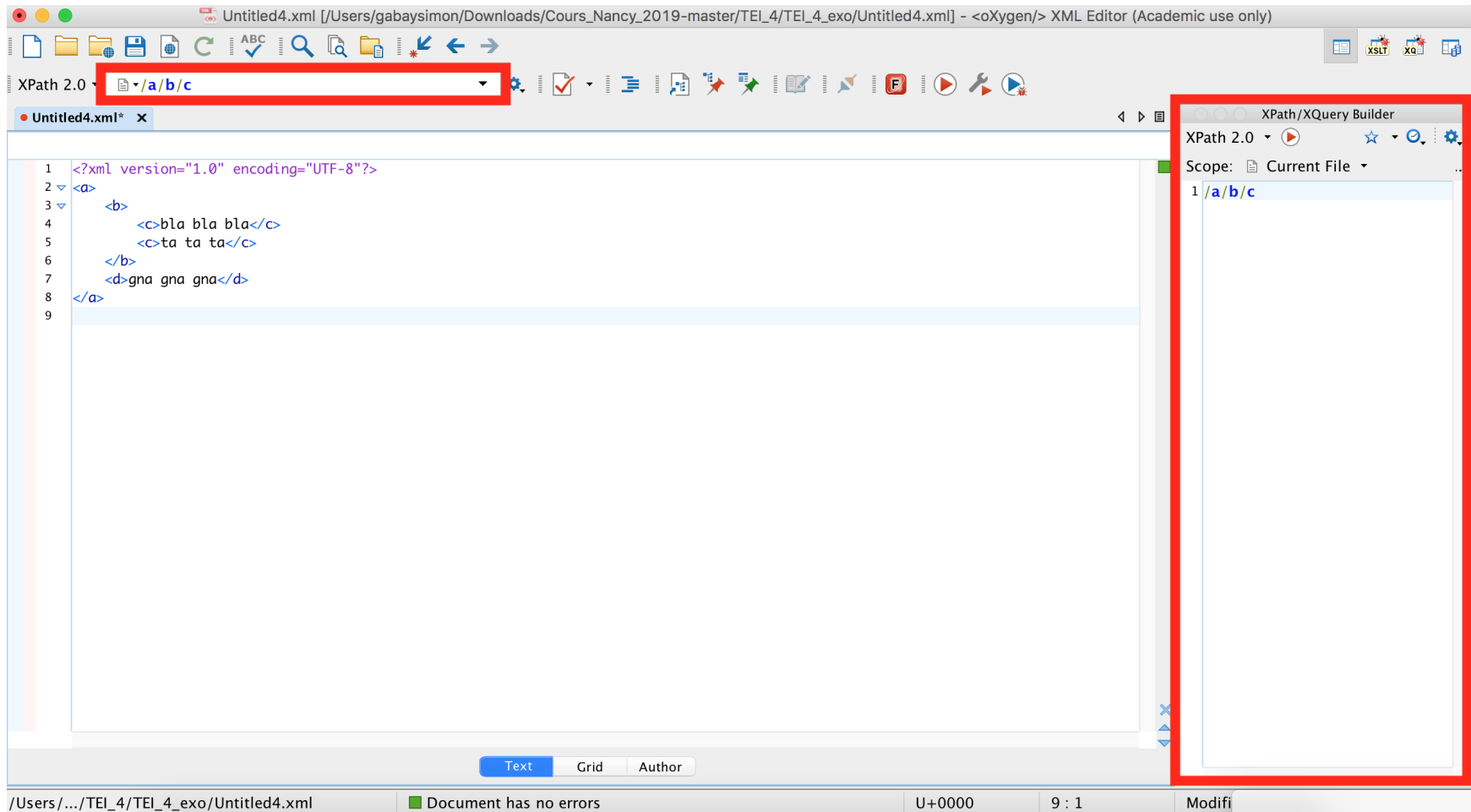
Pour les deux `<c>`, le chemin est le suivant:

```
/a/b/c
```

On reconnaît le `/` que nous avons vu précédemment

Dans Oxygen

On peut faire des requêtes XPath à deux endroits dans Oxygen:



Principes avancés: //

Il est possible de faire un raccourci:

```
/a/b/c
```

Peut s'écrire:

```
//c
```

Attention cependant: dans ce cas on récupère tous les éléments `<c>` .

Le résultat de nos deux requêtes XPath est donc différent pour le document XML suivant:

```
<a>
  <b>
    <c>bla bla bla</c>
    <c>ta ta ta</c>
  </b>
  <c>gna gna gna</c>
</a>
```

Principes avancés: les prédicats

Il est possible de préciser notre chemin avec des prédicats. Prenons le document suivant:

```
<a>
  <b>
    <c type="bla">bla bla bla</c>
    <c type="ta">ta ta ta</c>
  </b>
  <c>gna gna gna</c>
</a>
```

Il est possible de dire que l'on veut le premier `<c>` uniquement:

```
/a/b/c[1]
```

Ou que l'on veut un élément avec un attribut précis (attention aux guillemets simples:

```
/a/b/c[type='bla']
```

La famille xml

On décrit souvent le XML sur un mode généalogique (on parle "d'axe"):

```
<a>
  <b>
    <c type="bla">bla bla bla</c>
    <c type="ta">ta ta ta</c>
  </b>
  <c>gna gna gna</c>
</a>
```

Dans ce cas:

1. `` est le parent (*parent*) de `<c>`
2. `` est l'enfant (*child*) de `<a>`
3. `<c>` est le descendant (*descendant*) de `<a>`
4. `<a>` est l'ancêtre (*ancestor*) de `<c>`

Principes avancés: les axes

Il est possible d'utiliser ces informations généalogiques pour localiser des éléments précis:

```
<a>  
  <b>  
    <c>bla bla bla</c>  
    <c>ta ta ta</c>  
  </b>  
  <c>gna gna gna</c>  
</a>
```

```
//c[parent::a]
```

est l'équivalent de

```
/a/c
```

On peut combiner les prédicats:

```
//c[parent::b][1]
```