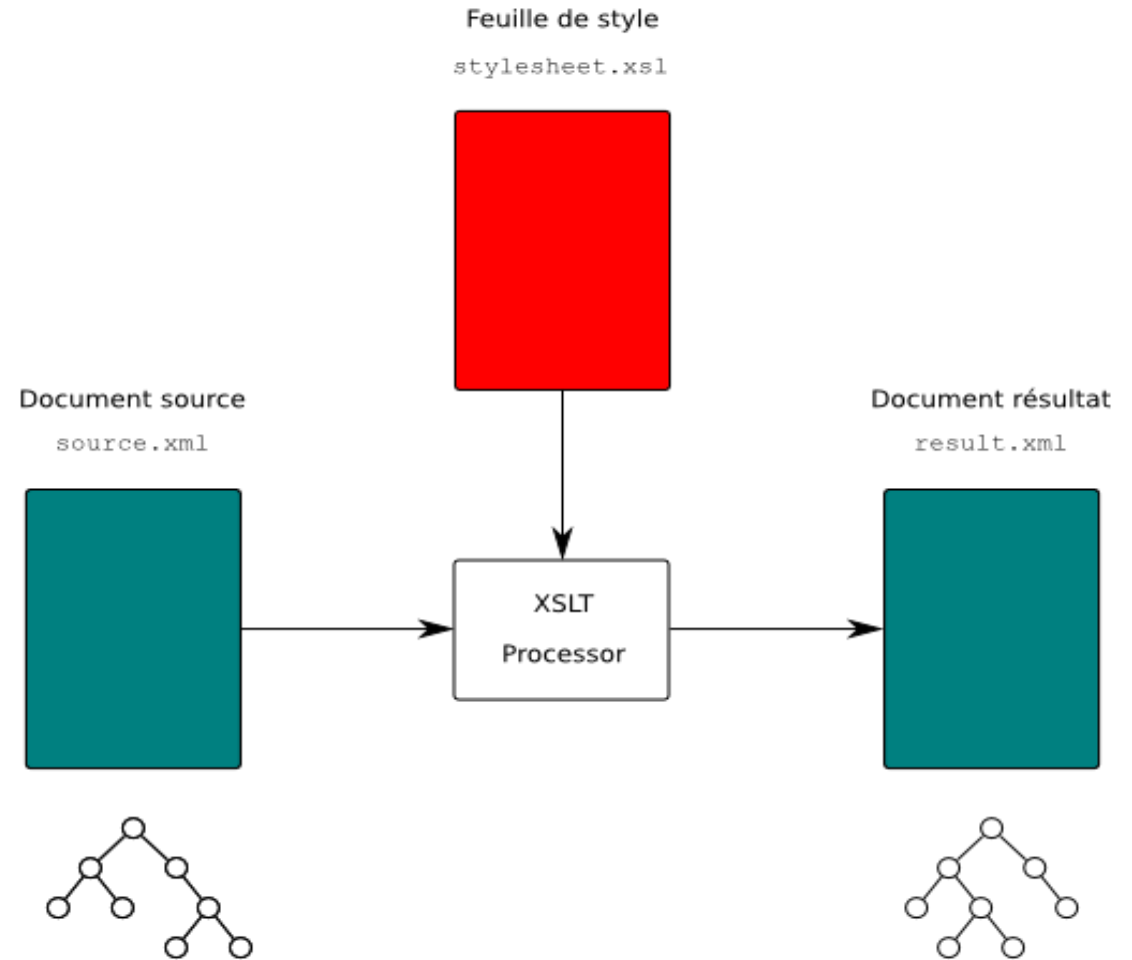


XSLT

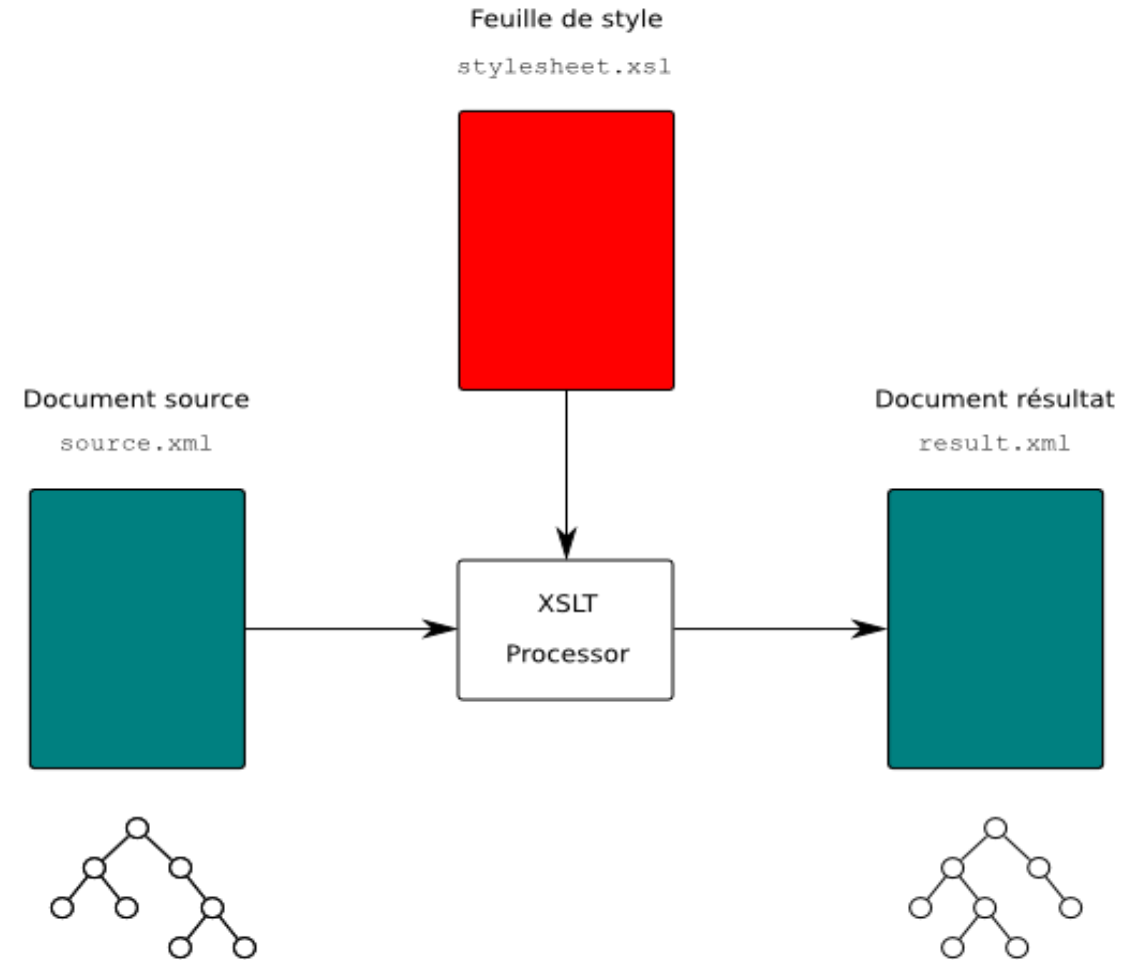
- **XSLT** (e**X**tensible **S**tylesheet **L**anguage **T**ransformations), ou langage extensible de feuille de style, est une technologie qui permet de transformer les informations d'un document **XML** vers un autre type de document: **XML, HTML, SVG, PDF, etc...**
- Le premier but de **XSLT** est de permettre l'affichage d'un document XML dans un navigateur.
- **XSLT** s'appuie sur **XPath** pour sélectionner des informations à transformer
- Le principe de fonctionnement est assez simple : un document **XSLT** est associé à un document **XML** afin de créer un nouveau document d'une nature différente ou identique.

- Un document **XSLT** est un document **XML**.
- On appelle souvent un document **XSLT** une **feuille de style XSL**
- Un **processeur XSLT** lit d'une part un document **XML** et d'autre part un document **XSLT** et génère un nouveau document en appliquant les règles de transformation.





le rôle de **XSLT** consiste à définir des règles de transformations qui vont produire un nouveau type de document (**XML** ou non).



Structure d'un document XSLT

- Le documents **XSLT** doit être écrit dans un fichier distinct du document **XML** dont l'extension est ".xsl".
- Une feuille **XSLT** doit commencer par ces lignes :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml"/>
```

- La deuxième ligne identifie la norme de la feuille de style et définit le **namespace xsl:** de ses éléments.
- La troisième ligne indique le type de sortie : **xml**, **html** ou **text**.
- Via la déclaration de cet **espace de noms**, toutes les balises utilisées dans un document **XSLT** doivent être préfixées par **xsl:**.

Structure d'un document XSLT

- Pour attribuer une feuille de style **XSL** à un document **XML**, il faut mettre ceci avant la racine du document:

```
<?xml-stylesheet type="text/xsl" href="FEUILLE.xsl"?>
```

- Cette balise possède 2 attributs
 - L'attribut **type**: permet de définir le type du document que nous souhaitons référencer. Dans notre cas, il s'agit d'un document XSLT
 - L'attribut **href**: permet d'indiquer l'**URI** du document que l'on souhaite référencer. Dans mon exemple, il s'agit d'un chemin relatif puisque le document **XML** et le document **XSLT** sont situés dans le même dossier.

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
    <company>CBS Records</company>
    <price>9.90</price>
    <year>1988</year>
  </cd>
</catalog>

```

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler

```

<?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"/>
  <xsl:template match="/">
    <html>
    <body>
      <h2>My CD Collection</h2>
      <table border="1">
        <tr bgcolor="#9acd32">
          <th style="text-align:left">Title</th>
          <th style="text-align:left">Artist</th>
        </tr>
        <xsl:for-each select="catalog/cd">
          <tr>
            <td><xsl:value-of select="title"/></td>
            <td><xsl:value-of select="artist"/></td>
          </tr>
        </xsl:for-each>
      </table>
    </body>
    </html>
  </xsl:template>
</xsl:stylesheet>

```

Corps d'un document XSLT.

- L'algorithme général de **XSLT** est :
 1. Il sélectionne (**match**) les éléments **XML** du fichier source.
 2. Pour chaque élément reconnu il génère une sortie sur le fichier cible
- Un document **XSLT** se compose d'un ensemble de règles (**template**) permettant de construire le résultat.
- Un modèle (**template**) de transformation est défini dans un élément **xsl:template**

Sources :

```
1 <a>
2   <b/><b/><c/>
3 </a>
```

Règle XSL-XSLT :

```
1 <xsl:template match='/a/b'>
2   BONJOUR
3 </xsl:template>
```

Résultat :

```
1 BONJOUR BONJOUR
```


- Chaque **règle (template)** est indépendante des autres et à en charge de sélectionner un élément dans la **source** et d'effectuer une écriture dans la **cible**.
- L'application d'une **règle** produit un fragment du document résultat.

Syntaxe : Structure générale d'un programme XSL-XSLT

```
1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3     <xsl:output method="html" indent="yes" encoding="iso-8859-1"/>
4     <xsl:template match="XPath">
5         ...
6     </xsl:template>
7     <xsl:template match="XPath">
8         ...
9     </xsl:template>
10    ...
11 </xsl:stylesheet>
```

- Les **templates** sont des sortes de couples (expression, contenu) : les parties du document XML qui correspondent à l'expression sont remplacées par le contenu.
 - L'expression est écrite en **XPath** et porte sur l'arbre **XML** d'entrée. Elle indique quels sont les nœuds à remplacer par le contenu.
 - Le contenu est un mélange d'éléments **XSL** et d'autres choses (éléments et textes) qui doivent respecter la syntaxe **XML**. *Syntaxe : Règle (template)*
-

```
1 <xsl:template match='XPATH'>
2     Instruction XSLT et/ou génération de texte sur la sortie
3 </xsl:template>
```

Instructions (balises) principales XSLT

1. **<xsl:template match='xpath'/'>** : la balise la plus importante du langage **XSLT**. C'est grâce à elle que l'on définit la transformation à effectuer. Cette balise permet de remplacer toutes les parties du document XML correspondantes à l'expression **XPath** par le contenu de la balise.

Ex: trouver et remplacer les éléments **student**?

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="class.xsl"?>
<class>
  <student>Adam</student>
  <student>Hachem</student>
  <student>Sirine</student>
  <teacher>Mr. Ahmed</teacher>
</class>
```

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl=
  "http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="student">
    Found a learner!
  </xsl:template>
</xsl:stylesheet>
```

➤ Le code **XSLT** trouvera les éléments **student** dans le document **XML** et produira «Found a learner! » pour chaque élément **student**. Cet exemple affiche à la fois le résultat et la fonctionnalité de l'ajout de texte de **XSLT**! (affiche « Found a learner! » trois fois, suivi d' *unmatched* texte de l'élément **teacher** "Mr. Ahmed".)

→ Found a learner! Found a learner! Found a learner! Mr. Ahmed

1. **<xsl:apply-templates select =''xpath''/>** : permet de continuer la transformation des éléments enfants d'un **template** (déclenche un appel récursif sur tous les fils du nœud courant). Le processeur **XSLT** va rechercher des **templates** correspondants et d'appliquer les transformations associées pour les éléments désignés par l'expression **xpath**. Avec cette balise, nous pouvons éliminer les *unmatched* textes en choisissant des éléments enfants spécifiques

Ex: trouver et remplacer les éléments **student**?

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="class.xsl"?>
<class>
  <student>Adam</student>
  <student>Hachem</student>
  <student>Sirine</student>
  <teacher>Mr. Ahmed</teacher>
</class>
```

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl=
  "http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <xsl:apply-templates select="/class/student"/>
  </xsl:template>
  <xsl:template match="student">
    Found a learner!
  </xsl:template>
</xsl:stylesheet>
```

1. Le premier **template** à être appelé est le **template** dont l'expression **XPath** capture la racine. Dans ce **template**, la ligne **<xsl:apply-templates select="/class/student"/>** permet d'indiquer que l'on souhaite continuer la transformation uniquement avec **<student>** ainsi que ses fils
2. L'élément **<student>** va donc être transformé grâce au second **template** puisque son expression XPath le capture. ➔ Found a learner! Found a learner! Found a learner!

Ex2:

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "students.xsl"?>
<class>
  <student id = "393">
    <firstname>Dinkar</firstname>
    <lastname>Kad</lastname>
  </student>
  <student id = "493">
    <firstname>Vaneet</firstname>
    <lastname>Gupta</lastname>
  </student>
  <student id = "593">
    <firstname>Jasvir</firstname>
    <lastname>Singh</lastname>
  </student>
</class>
```

Students

ID:393
First Name:Dinkar
Last Name:Kad

ID:493
First Name:Vaneet
Last Name:Gupta

ID:593
First Name:Jasvir
Last Name:Singh

```

<xsl:template match = "/">
  <html>
    <body>
      <h2>Students</h2>
      <xsl:apply-templates select = "class/student" />
    </body>
  </html>
</xsl:template>
<xsl:template match = "class/student">
  <xsl:apply-templates select = "@id" />
  <xsl:apply-templates select = "firstname" />
  <xsl:apply-templates select = "lastname" /><br />
</xsl:template>
<xsl:template match = "@id">
  ID:<span style = "color:red;font-size = 22px;">
    <xsl:value-of select = "." />
  </span> <br />
</xsl:template>
<xsl:template match = "firstname">
  First Name:<span style = "color:blue;">
    <xsl:value-of select = "." />
  </span> <br />
</xsl:template>
<xsl:template match = "lastname">
  Last Name:<span style = "color:green;">
    <xsl:value-of select = "." />
  </span> <br />
</xsl:template>

```


1. **<xsl:value-of select = 'xpath' />** : permet d'extraire la valeur d'un élément **XML** ou la valeur de ses attributs. Elle possède un attribut **select** auquel il convient de renseigner une expression **xpath** permettant de sélectionner les informations à extraire

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="classValue.xsl"?>
<class>
  <student id="1">Adam</student>
  <student id="2">Hachem</student>
  <student id="3">Sirine</student>
  <teacher>Mr. Ahmed</teacher>
</class>
```

1 - Adam

2 - Hachem

3 - Sirine

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl=
  "http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="class">
    <html>
      <body>
        <xsl:apply-templates select="student"/>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="student">
    <p>
      <xsl:value-of select="@id"/> -
      <xsl:value-of select="."/>
    </p>
  </xsl:template>
</xsl:stylesheet>
```

1. **<xsl:for-each select = "xpath">** : Le contenu de cette balise sera exécuté une fois pour tous les éléments qui correspondent à l'expression XPath.

```
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
  </cd>
  <cd>
    <title>Greatest Hits</title>
    <artist>Dolly Parton</artist>
  </cd>
  <cd>
    <title>Still got the blues</title>
    <artist>Gary Moore</artist>
  </cd>
  <cd>
    <title>Eros</title>
    <artist>Eros Ramazzotti</artist>
  </cd>
</catalog>
```

```
<xsl:for-each select="XPATH">
  CONTENU
</xsl:for-each>
```

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti

```
<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```



1. **<xsl:sort />** : permet de trier un ensemble de nœuds, résultant de **<xsl:apply-templates>** ou **<xsl:for-each>**, sur un critère donné grâce à l'attribut **select**, qui contient une expression **xpath** relative au nœud courant. Avec l'attribut **order**, on peut choisir l'ordre du tri, **ascending** ou **descending**.

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti



My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Eros	Eros Ramazzotti
Greatest Hits	Dolly Parton
Hide your heart!	Bonnie Tyler
Still got the blues	Gary Moore

```
<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
      <xsl:sort select="title" order="ascending"/>
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```



1. `<xsl:if test="condition" />` : l'instruction **if** permet d'appliquer une transformation uniquement si la **condition** est vérifiée.


```
<xsl:if test="CONDITION">
    CONTENU
</xsl:if>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="classValue.xsl"?>
<class>
  <student id="1">Adam</student>
  <student id="2">Hachem</student>
  <student id="3">Sirine</student>
  <teacher>Mr. Ahmed</teacher>
</class>
```

1 - Adam

3 - Sirine

```
<xsl:template match="class">
  <html>
  <body>
    <xsl:apply-templates select="student"/>
  </body>
</html>
</xsl:template>
<xsl:template match="student">
  <p >
    <xsl:if test="@id!=2">
      <xsl:value-of select="@id"/> -
      <xsl:value-of select="."/>
    </xsl:if>
  </p>
</xsl:template>
</xsl:stylesheet>
```



1. **<xsl:choose>** : permet de réaliser plusieurs tests. Il contient des éléments **xsl:when** et éventuellement un élément **xsl:otherwise**. Chacun des éléments **xsl:when** possède un attribut **test** contenant une expression **XPath** servant de **condition**.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="classChoose.xsl"?>
<class>
  <student id="1" sexe="1">Adam</student>
  <student id="2"  sexe="1">Hachem</student>
  <student id="3" sexe="2">Sirine</student>
  <student id="3">Samir</student>
  <student id="3" sexe="3">Amine</student>
  <teacher>Mr. Ahmed</teacher>
</class>
```

masculin

masculin

féminin

sexe non spécifié

code de sexe inconnu

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl=
  "http://www.w3.org/1999/XSL/Transform">
<xsl:template match="class">
  <html>
  <body>
    <xsl:apply-templates select="student"/>
  </body>
</html>
</xsl:template>
<xsl:template match="student">
  <xsl:choose>
    <xsl:when test="@sexe='1'"> <p> masculin </p>
    </xsl:when>
    <xsl:when test="@sexe='2'"><p> féminin </p>
    </xsl:when>
    <xsl:when test="not(@sexe)"><p> sexe non spécifié </p>
    </xsl:when>
    <xsl:otherwise> <p> code de sexe inconnu</p>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
</xsl:stylesheet>
```