

# Introduction aux langages XML : eXtensible Markup Language et XSL : eXtensible Stylesheet Language



# eXtensible Markup Language : Le concept

- Un fichier html est un fichier à balise qui contient de l'information et la façon d'afficher cette information à l'aide d'un browser (ie, netscape, mozilla, ...).
- *Le concept XML vise essentiellement*
  - *à séparer l'information de sa présentation*
  - *à pouvoir accéder facilement à toute ou partie de cette information*
  - *structurer l'information sous forme d'arbre*



# Les éléments d'un arbre XML

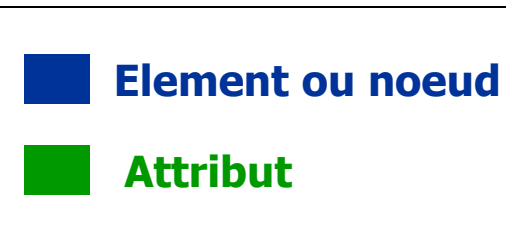
- Un arbre XML est un ensemble de nœuds délimités par une balise ouvrante et la même balise fermante.
- Une balise ouvrante à un nom et contient de 0 à plusieurs attributs et est définie comme suit :  
`<nomBalise att1= " ... " ... attn= " ... " >.`
- Une balise fermante se définit comme suit :  
`</nomBalise>`
- Un attribut a un nom et une valeur
- Un nœud contient de 0 à plusieurs nœuds et de 0 à plusieurs zones de textes (PCDATA) appelés nœuds textes.
- On peut définir des nœuds vides comme suit :  
`<nomBalise listeAttributs/>`. La liste des attributs peut être vide.



# XML : Un exemple de fichier XML

```
<etudiant prenom="Maude" nom="Eme" annee="ing2">
  <ou adresse="1 rue du bo" ville="Paris" cp="75000" />
  <matiere libelle="info">
    <appreciation>tu peux mieux faire</appreciation>
    <note epreuve="ds n° 1" valeur="10.5" />
    <note epreuve="ds n° 2" valeur="11" />
    <note epreuve="ds n° 3" valeur="12" />
  </matiere>
  <matiere libelle="comptabilité générale">
    <appreciation>belle prestation</appreciation>
    <note epreuve="ds" valeur="15" />
  </matiere>
</etudiant>
```

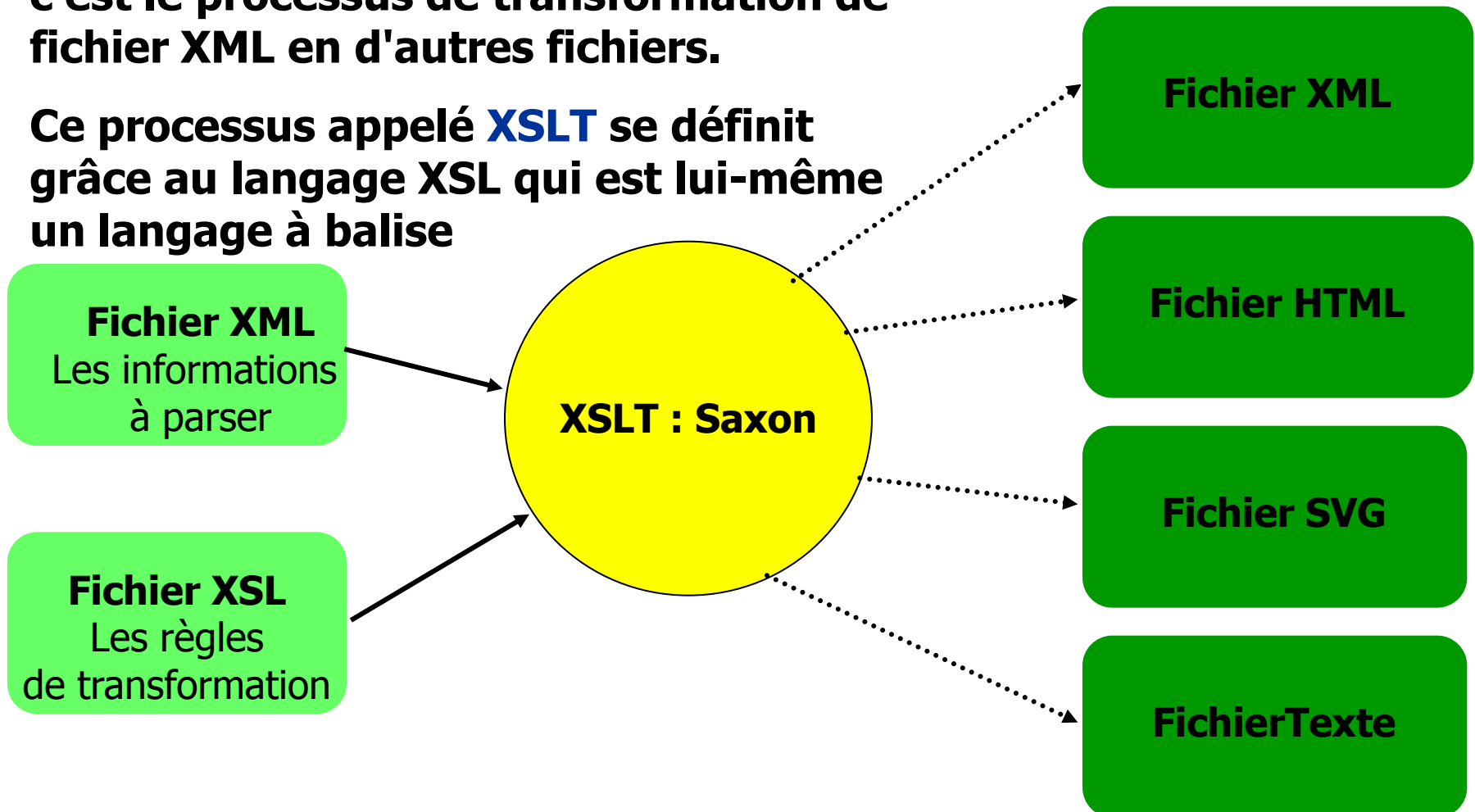
Vers une transformation  
en un fichier HTML



# XSL : eXtensible Stylesheet Language

Ce qui nous intéresse dans ce cours, c'est le processus de transformation de fichier XML en d'autres fichiers.

Ce processus appelé **XSLT** se définit grâce au langage XSL qui est lui-même un langage à balise



# L'entête d'un fichier XSL

- Un fichier XSL est un fichier XML dont la racine est définie par la balise `xsl:stylesheet`.
- Ce fichier définit des règles pour générer une sortie. On utilise le nœud fils `xsl:output` pour définir le type de la sortie.

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

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

```
<xsl:output method="html" indent="yes"  
  encoding="iso-8859-1" />
```

- L'attribut `xmlns:xsl` indique que l'on va utiliser dans la suite, des balises de la forme `xsl:...`. Dans ce cas `xsl` indique un espace de noms (vu plus tard).



# XSL : Un exemple de source XSL

```
<!-- le patron du nœud etudiant -->
<xsl:template match="etudiant">
<html>
  <body>
    <h1>
      <xsl:value-of select="@prenom" />
      <xsl:text> </xsl:text>
      <xsl:value-of select="@nom" />
    </h1>
    <xsl:apply-templates select="ou" />
  </body>
</html>
</xsl:template>
```



# XSL : Un exemple de source XSL

```
<!-- le patron du nœud ou -->  
<xsl:template match="ou">  
<h2>  
<xsl:value-of select="@adresse" /><br />  
<xsl:value-of select="@cp" />  
<xsl:text> </xsl:text>  
<xsl:value-of select="@ville" />  
</h2>  
</xsl:template>  
<!-- fin de la feuille xsl-->  
</xsl:stylesheet>
```





# XSLT : Un exemple de transformation

- Le fichier ci-dessous est la transformation XSLT des fichiers étudiés précédemment.

```
<html>
```

```
  <head>
```

```
    <meta http-equiv="Content-Type" content="text/html;  
    charset=iso-8859-1">
```

```
  </head>
```

```
  <body>
```

```
    <h1>Maude Eme</h1>
```

```
    <h2>
```

```
      1 rue du bo<br>75000 Paris
```

```
    </h2>
```

```
  </body>
```

```
</html>
```

Vers le fichier XML

Vers le fichier XSL



# Le langage XSL

- Un source XSL est un fichier XML composé :
  - D'une entête
  - De patrons de nœuds composés eux-mêmes
    - De paramètres et variables locales
    - De structures conditionnelles
    - De structures d'aiguillages
    - De structures itératives portant sur les nœuds fils
  - De paramètres globaux



# XSL : les paramètres et les variables

- On peut définir des paramètres avec l'instruction
  - `<xsl:param name="nomParam">...</xsl:param>`
- On peut définir des variables locales à un noeud avec l'instruction
  - `<xsl:variable name="nomVar">...</xsl:variable>`
- On peut récupérer le contenu d'une variable ou d'un paramètre avec l'expression `$nomElement` où `nomElement` est le libellé de la variable ou du paramètre.



# L'ordre xsl:value-of

L'ordre `<xsl:value-of select= " ??? " / >`

- Cet ordre envoie dans la sortie le texte qui est contenu dans la zone indiquée par ???.
- Si ??? est un nœud, alors tous les nœuds textes de l'arbre sont concernés par l'envoi.
- Si ??? est une variable ou un paramètre, alors le contenu texte est concerné par l'envoi.
- Si ??? est une expression, alors l'évaluation de l'expression est concernée par l'envoi.



# XSL : Patron de nœud

- L'instruction `xsl:template` permet de définir pour un nœud ce que le processeur XSLT devra faire quand il devra traiter ce nœud.
- La syntaxe est  
`<xsl:template match="NomDuNoeud">`  
....  
`</xsl:template>`
- Si un patron de nœud n'est pas défini, il vaut par défaut :  
`<xsl:template match="nomDuNoeud">`  
    `<xsl:apply-templates />`  
`</xsl:template>`



# XSL : L'appel d'un patron de nœud

- L'instruction `xsl:apply-template` permet de préciser au processeur XSLT quel patron de nœud il faut appliquer à un endroit précis du fichier XML où ce nœud peut être un fils du nœud courant.
- La syntaxe est
  - `<xsl:apply-templates select="NomDuNoeudFils" />`
  - `<xsl:apply-templates />` pour tous les nœuds fils.



# XSL : L'appel d'un patron de nœud

- L'instruction `xsl:apply-template` permet au concepteur de la feuille XSL de gérer les appels des fils du nœud courant.
- En particulier, des nœuds fils peuvent être dans un certain ordre dans le fichier XML et être traités dans un autre ordre ou ne pas être traités.



# XSL : La clause select

- Cette clause est utilisée dans les instructions `xsl:apply-template` et `xsl:for-each`. Elle porte sur des nœuds.
- On peut sélectionner ces nœuds avec une condition en utilisant la syntaxe suivante

`select="nomDuNoeud[condition]"`





# XSL : Les tests

- L'instruction `xsl:if` permet de préciser au processeur XSLT un traitement conditionnel.
- La syntaxe est  
`<xsl:if test="..." />`  
...  
`</xsl:if>`



# XSL : Les tests

- La condition définie entre guillemets peut porter
  - Sur l'existence ou non de nœuds fils, l'existence ou non de variables, de paramètres ou d'attributs.
  - Sur le contenu de nœuds fils, de variables, de paramètres ou d'attributs.



# XSL : Les tests

- Lors de l'exécution du processeur XSLT, on appellera **contexte** l'arbre défini par le nœud courant.
- La fonction **count(node-set)** renvoie le nombre de nœuds de l'ensemble de nœuds node-set passé en argument.
- La fonction **position()** renvoie la place du nœud courant dans le contexte de son père.



# XSL : Les tests

- La fonction `last()` renvoie la taille du contexte du père du nœud courant.
- Les opérateurs booléens sont `or`, `and` et `not(...)`
- Les opérateurs de comparaison `=`, `&lt;`, `&gt;`, `!=`
- La fonction `node()` indique tous les nœuds fils du nœud courant



# XSL : Les tests

<!-- on teste si le nœud courant est le premier fils de son père-->

<xsl:if test="position() = 1">

<!-- on teste si le nœud courant est le dernier fils de son père-->

<xsl:if test="position() = last()">

<!-- on teste si la variable v1 est inférieure v2 -->

<xsl:if test="\$v1 < \$v2">



# XSL : Les tests

<!-- on teste si le nœud courant à un fils  
nommé theFils-->

<xsl:if test="theFils">

<!-- on teste si le nœud courant à un attribut  
nommé theAttribut-->

<xsl:if test="@theAttribut">



# XSL : Les tests

<!-- on teste si le nœud courant à un attribut nommé theAttribut qui vaut val -->

```
<xsl:if test="@theAttribut = 'val' ">
```

<!-- on teste si le nœud courant à une variable ou un paramètre nommé theParVal qui vaut val -->

```
<xsl:if test="$theParVal = 'val' ">
```



# XSL : Les aiguillages

- L'instruction `<xsl:choose>` permet de gérer des branchements conditionnels à plusieurs branchements
- La syntaxe est

```
<xsl:choose>
```

```
  <xsl:when test="...">...</xsl:when>
```

```
  ...
```

```
  <xsl:otherwise>...</xsl:when>
```

```
</xsl:choose>
```





# XSL : les attributs

- On peut accéder à un attribut d'un nœud (ou élément) avec l'expression `@nomAttribut`.
- On peut définir des listes d'attributs

```
<xsl:attribute-set name="nomListeAtt">  
  <xsl:attribut name="nomAtt1">...</xsl:attribut>  
  ...  
  <xsl:attribut name="nomAttn">...</xsl:attribut>  
</xsl:attribute-set>
```
- Une liste d'attributs sert au processeur XSLT à fabriquer les attributs d'une balise.
- La syntaxe est

```
<nomBalise xsl:use-attribute-set="nomListeAtt" ..>
```

