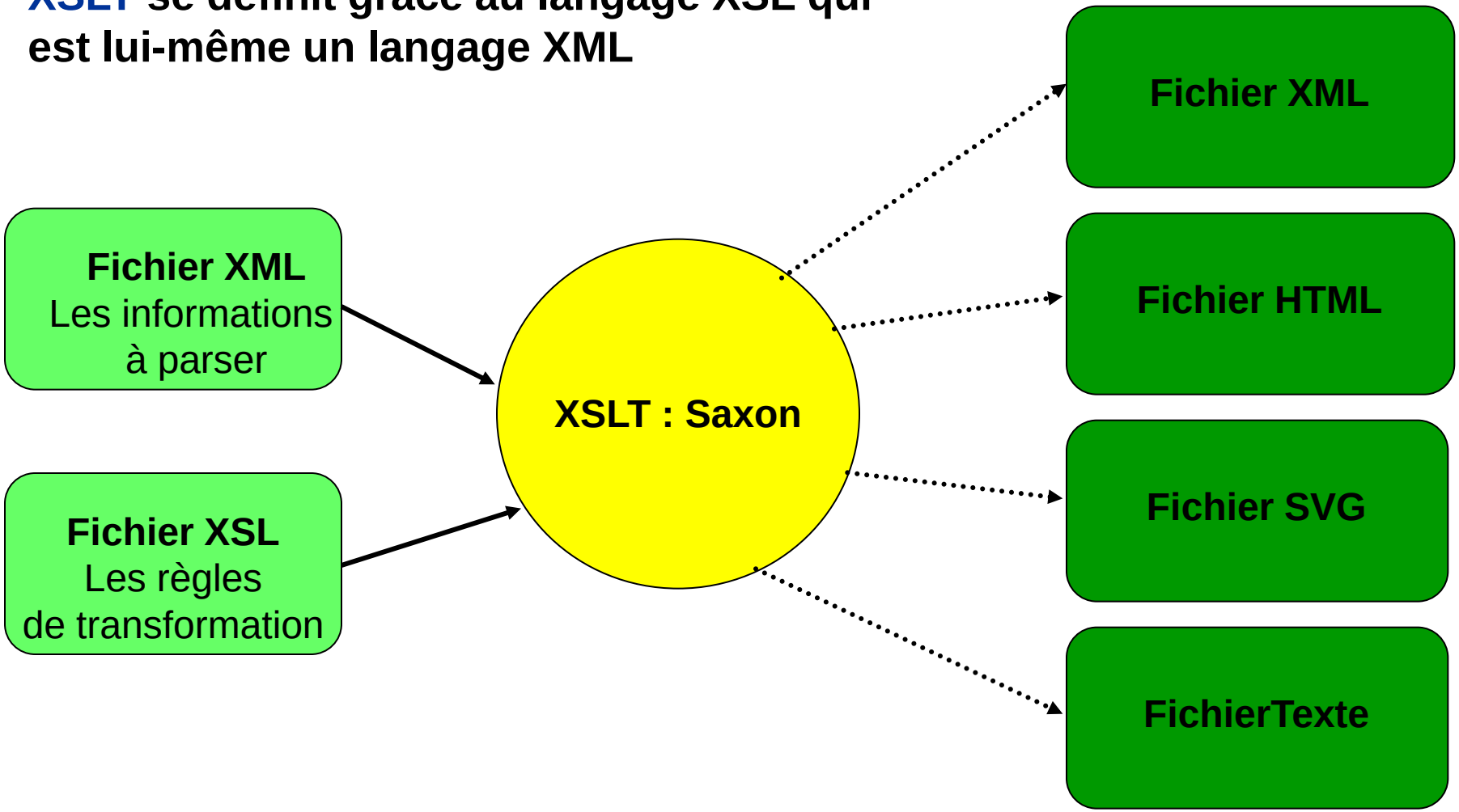
EISTI
Ecole Internationale des Sciences du Traitement de l'Information

XML +XSL

Compléments en XSLT 2.0

Le processus XSLT

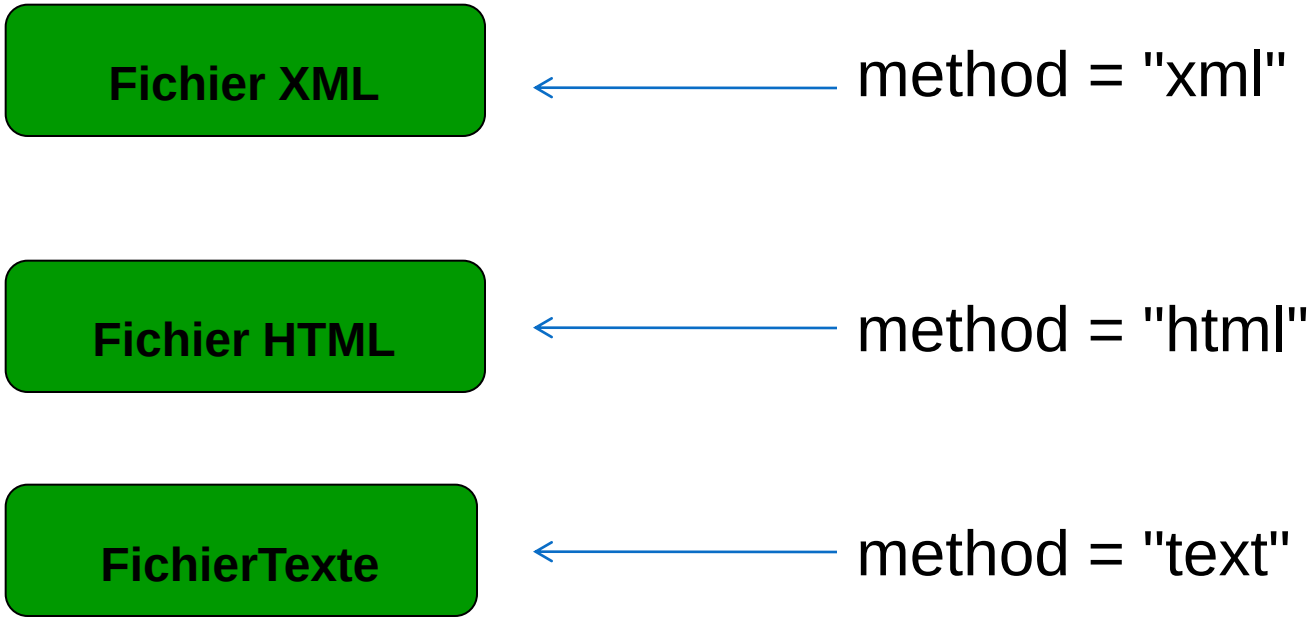
Le processus de transformation appelé XSLT se définit grâce au langage XSL qui est lui-même un langage XML



L’entête d’une feuille XSL

```
<?xml version="1.0" encoding= "utf-8"?>
<xsl:stylesheet version= " 2.0"  xmlns:xsl= "http://www.w3.org/1999/XSL/Transform">

<xsl:output method="???" version="1.0"  encoding= " utf-8" indent="yes"/>
```



Les espaces de noms

- Dans un fichier XML, on peut être amené à mélanger plusieurs langages XML.
- On peut donc avoir des ambiguïtés de noms de balises.
- Pour résoudre ce problème, on a inventé les espaces de noms.
- Solution : les balises d'un langage XML peuvent être préfixées
- Dans ce cas le préfixe et le suffixe d'une balise sont séparés par le symbole ":" . Ce préfixe s'appelle un espace de noms.

Les espaces de noms

- Le langage XSL a pour espace de noms "xsl"
- Un espace de noms est identifié par un identifiant uniforme de ressources appelé URI de l'espace de noms
- L'URI du langage XSL est http://www.w3.org/1999/XSL/Transform
- Dans une feuille XSL, quand on utilise un langage préfixé il faut le préciser dans un attribut de la balise xsl:stylesheet.
- Cet attribut est de la forme xmlns:prefixe="valeur de l'URI"
- Cet URI est en général une URL mais il n'est pas obligatoire que cette URL pointe vers un document

Une feuille XSL et plusieurs sorties

- Par défaut, une exécution XSLT génère une seule sortie. Cette sortie est la sortie standard.
- La balise xsl:result-document permet d'avoir une sortie sur un fichier.
- Cette balise a trois attributs :
  - format avec les valeurs "html", "xml" et "text"
  - encoding pour préciser l'encodage du fichier
  - href pour préciser le chemin du fichier
- Cette balise peut être utilisée plusieurs fois dans une feuille XSL. On peut donc avoir plusieurs sorties
- A l'intérieur du nœud, on précise ce qui doit être envoyé dans le fichier.

Chargement dynamique de fichiers XML

- Une transformation XSLT a deux paramètres : un fichier XML et un fichier XSL
- En plus du fichier XML, on peut charger à tout moment de l'exécution d'autres fichiers XML
- On utilise dans un attribut select la fonction document.
- L'instruction xsl est la suivante :

  <xsl:value-of select="document(nomDuFichier)"/>
- Le nom du fichier peut être paramétré dans la feuille xsl
- En général, cette instruction est utilisée dans une variable. Puis on utilise cette variable en la parcourant en tant qu'arbre XML

xsl:include et xsl:import

- Dans une feuille XSL, on peut inclure une autre feuille XSL
- Deux possibilités : xsl:include et xsl:import
- ATTENTION : un fichier ne peut pas s'inclure lui-même directement ou indirectement
- Syntaxe :
  - <xsl:include href="???"/>
  - <xsl:import href="???"/>

xsl:include et xsl:import

- xsl:include
  - On inclut à n'importe quel endroit de la feuille XSL et on peut inclure plusieurs feuilles XSL.
  - Le processeur construit une feuille XSL en incluant tous les fichiers. Si dans ce fichier un template est répété alors le processeur s'arrête en générant une erreur.
- xsl:import
  - On inclut impérativement juste après la balise xsl:stylesheet.
  - Un template peut être répété. Dans ce cas, le processeur prend le premier rencontré

Fonctions

- Les templates nommés peuvent être appelés directement comme une fonction dans un langage procédural
- L'appel d'un tel template est très lourd par rapport à une fonction dans un langage procédural
- Le langage XSL fournit des fonctions prêtes à l'emploi : voir http://www.w3.org/TR/xpath-functions/
- On peut définir des fonctions. Un exemple ci-dessous

```
<xsl:function name="fonc:somme" as="xsd:double">
    <xsl:param name="ope1"/>
    <xsl:param name="ope2"/>
    <xsl:value-of select="$ope1 + $ope2"/>
</xsl:function>
```

Fonctions

- On définit une fonction dans un nœud xsl:function.
- Ce nœud a deux attributs :
  - name="???" pour le nom de la fonction
  - as = "???" pour le type de retour de la fonction
- Pour le retour, on peut utiliser tous les types définis dans le langage xsd permettant de définir un schéma XML.
- Les noms des fonctions doivent être préfixés par un espace de noms.
- Une fonction s'appelle dans un attribut select comme dans un langage procédural classique

```
<xsl:value-of select="fonc:somme(1,2)"/>
```

xsl:for-each-group

- L'instruction xsl:for-each permet de parcourir un ensemble d'éléments un par un
- L'instruction xsl:for-each-group permet de partitionner un ensemble d'éléments et de parcourir partie par partie.
- Cette balise a deux attributs :
  - select = "???" qui définit l'ensemble de tous les éléments
  - group-by = "???" qui définit le critère de partition
- A chaque itération de for-each-group :
  - La fonction current-grouping-key() renvoie la valeur commune du critère de partition de la partie courante.
  - La fonction current-group () renvoie l'arbre XML de la partie courante.

Template : mode

- Dans un nœud template, on peut avoir l'attribut (optionnel) mode.
- Cela permet de traiter un nœud de plusieurs façons différentes en fonction du contexte
- Exemple : dans un xml qui décrit un texte structuré avec le nœud entete. On voudra traiter ce nœud pour l'afficher dans la table des matières et faire un autre traitement pour l'afficher dans un paragraphe
- Syntaxe d'appel :

```
<xsl:apply-templates select="???" mode="???">
<xsl:apply-templates mode="???">
```

Template : priority

- Dans un nœud template, on peut avoir l'attribut (optionnel) priority.
- Cet attribut permet au processeur XSLT d'appeler le bon template quand il a le choix entre plusieurs
- Des cas de figures
  - Un template avec match="*" sera toujours en conflit avec un template avec match="nom"
  - Un template (1) dont le match utilise le symbole | sera en conflit avec un template (2) dont la valeur de match est un des noms utilisé dans le template (1).
  - On peut avoir deux versions d'un même template à cause d'un xsl:import.
- Pour régler le conflit, vous pouvez définir une priorité en donnant une valeur réelle avec l'attribut priority ou vous appuyez sur les priorités par défaut : slide suivant

Template : priority

Priorité par défaut	Format de modèle de chemin d'accès	Exemples
- .5	Nom essai générique (un nom), ou test de type de nœud (peu importe le nom)	* @ * noeud () texte () comment () processing-instruction ()
- .25	Joker espace de noms qualifié (quel que soit le nom local)	xyz: * @ Xyz: *
0	test de nom pour un nom particulier, ou processing-instruction (*littérale*)	foo xyz: foo @ Foo @ Xyz: foo processing-instruction ('foo')
.5	Tout autre modèle de chemin d'accès. En d'autres termes, toute celle qui comprend l'un de ces opérateurs: / , / / ou []	/ / Foo foo / bar foo [2] / Foo [bar = 'bat']

Séquence

- On peut créer une séquence de nœuds ou de valeurs atomiques

```
<xsl:sequence select="1 to 5"/>
```

-
```
<xsl:variable name="seq" as="xs:integer *">
    <xsl:for-each select="1 to 5">
        <xsl:sequence select=". * ."/>
    </xsl:for-each>
</xsl:variable>
```

-
```
<xsl:variable name="moyennes" as="xsd:double *">
<xsl:sequence select="lyceen/moyenne"/>
</xsl:variable>
```

- Une séquence stockée dans une variable peut être parcourue avec un xsl:for-each