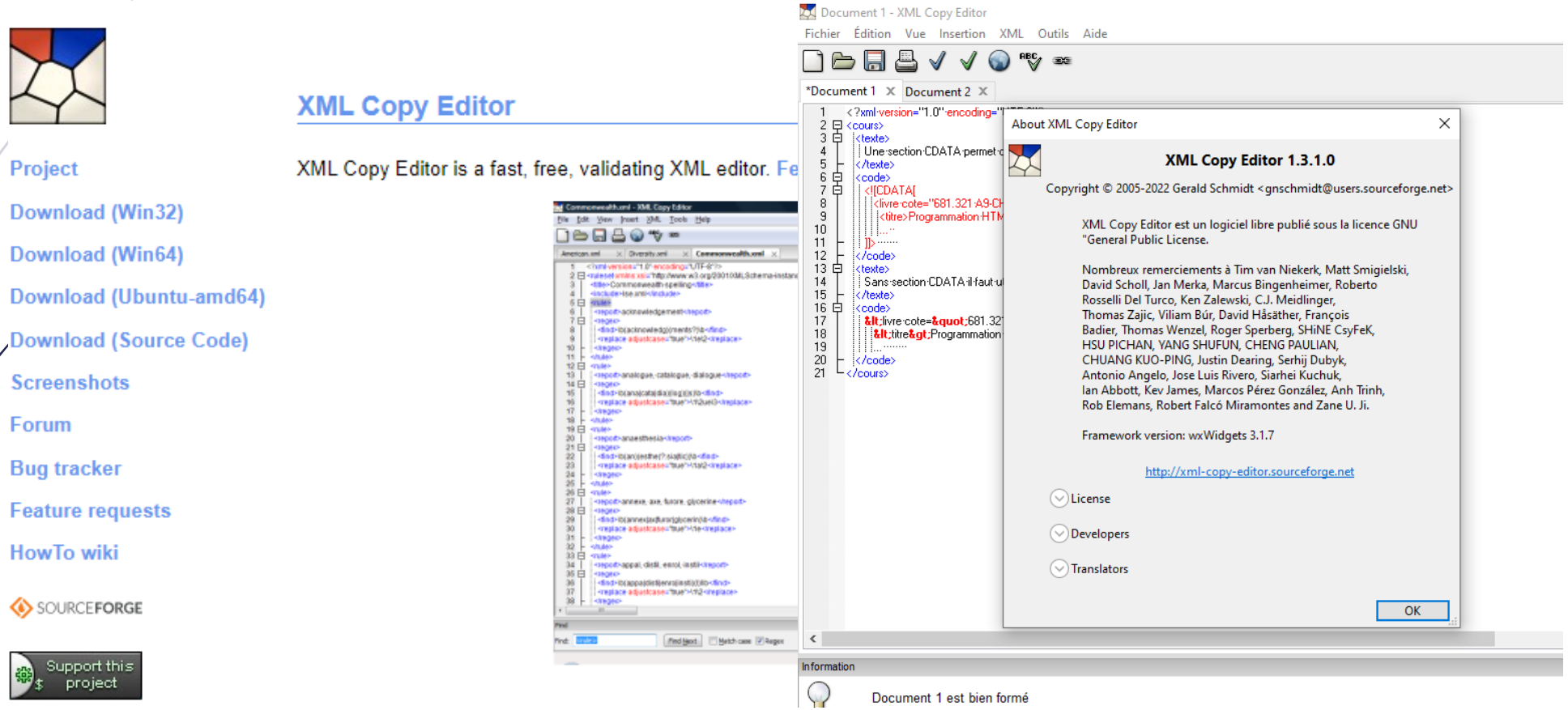
مدرسة علوم المعلومات
ⵜⵉⵎⵔⵔⴰⵜ ⵏ ⵜⵓⵙⵙⵏⵉⵡⵉⵏ ⵏ ⵜⵣⵔⴼⵜ
ECOLE DES SCIENCES
DE L'INFORMATION

Elément 02 :

Documents structurés

Pr . J. IDRAIS

Travaux Pratiques



Lien logiciel http://xml-copy-editor.sourceforge.net

Chapitre 4
Xpath & XSLT

Le langage XPATH

XPath est un langage d'interrogation des documents XML. Il permet de sélectionner certaines parties d'un document XML : des sous-arbres, des noeuds, des attributs, etc.

XPath est central dans le monde XML, il intervient comme brique de base dans d'autres technologies XML :

- les transformations XSLT,
- XQuery
- XLink
- XPointer,
- etc.

Le langage XPATH

- Syntaxe et sémantique partagées par d'autres outils (XSLT, Xpointer…)
- **But** : accéder aux différentes parties d'un document XML
- Représentation en forme d'arbre
- Type de nœuds :

  Racine, éléments, texte, attributs, espace de noms, instruction de traitement, commentaires

XPath : syntaxe générale

Le premier concept est celui de **noeud courant** : c'est l'endroit d'où l'on part. En première lecture on peut imaginer qu'il s'agit de la racine du document, mais n'importe quel noeud peut jour ce rôle.

```
axe1::filtre1[prédicat1]/axe2::filtre2[prédicat2]
```

Exemple concret :

```
parent::*/child::node()[position()=2]
```

XPath : syntaxe générale

```
axe1::filtre1[prédicat1]/axe2::filtre2[prédicat2]
```

Exemple concret :

```
parent::*/child::node()[position()=2]
```

À partir de là, on considère trois éléments :

- **un axe** : la direction dans laquelle on se dirige à partir du noeud courant (vers le père, vers les fils, vers les frères de gauche, etc.) ;
- **un filtre** : le type de noeuds qui nous intéresse dans l'axe choisi (des noeuds quelconques, des éléments quelconques ou un élément précis, des commentaires, etc.) ;
- **un prédicat optionnel** : des conditions supplémentaires pour sélectionner des noeuds parmi ceux retenus par le filtre dans l'axe.

XPath : syntaxe générale

Les axes

- **self** : le noeud courant lui-même ;
- **child** : les enfants du noeud courant ;
- **descendant**, **descendant-or-self** : tous les descendants du noeud courant ;
- **parent** : le père du noeud courant ;
- **ancestor**, **ancestor-or-self** : les ancêtres du noeud courant ;
- **attribute** : les attributs du noeud courant ;
- **preceding**, **following** : les noeuds, précédants ou suivants, du noeud courant, dans l'ordre de lecture du document ;
- **preceding-sibling**, **following-sibling** : les frères, précédant ou suivant, le noeud courant ;
- **namespace** : les espaces de noms.

XPath : syntaxe générale

Les filtres

- **node()** : tous les noeuds ;
- **text()** : les noeuds textuels ;
- ***** : tous les éléments ;
- **nom** : les éléments portant ce nom ;
- **comment()** : les noeuds commentaires ;
- **processing-instruction('cible')** : les noeuds instructions, seulement les instructions cible si cet argument est fourni.

XPath : syntaxe générale

Les prédicats

prennent la forme de tests que les nœuds sélectionnés devront vérifier. Ces tests peuvent impliquer des fonctions ou de nouveaux chemins XPath.

```
child::livre[position()=20]

child::livre[@ref='250']
```

Il est possible de combiner ces tests à l'aide des opérateurs logiques classiques (and, or et not) ou de les enchaîner :

```
child::livre[@ref='CS9000938125' or @ref=‘CS800008247']

child::livre[contains(text(),'XML') and position()=102]

child::livre[contains(text(),‘C++')][position()=205]
```

XPath : syntaxe générale

Les fonctions

Ces fonctions peuvent apparaître dans des prédicats ou être utilisées directement dans un évaluateur d'expressions XPath.

Il y a des fonctions sur les chaînes de caractères et qui vont donc porter sur les contenus textuels de noeuds :

Fonction	signification
concat	colle ensemble les chaînes de caractères passées en paramètres ;
string-length	la longueur de la chaîne fournie ;
contains ***starts-with*** ***ends-with***	tests d'appartenance d'une chaîne dans une autre.
count	le nombre de noeuds dans l'ensemble sélectionné par la requête ;
name	le nom de l'élément courant.
last	le nombre de noeuds sélectionnés à l'étape courante.
position	le numéro du noeud courant dans la liste des noeuds considérés ;

XPath : syntaxe générale

La syntaxe abrégée

Cette notation est plus simple mais pas aussi expressive que la notation étendue. De plus, l'équivalent étendu de certaines notations abrégées n'est pas toujours celui que l'on pense.

SYNTAXE ABRÉGÉE	SYNTAXE ÉTENDUE
`.`	`self::node()`
`auteur`	`child::auteur`
`../auteur`	`parent::auteur`
`@id`	`attribute::id`
`//auteur`	`/descendant-or-self::node()/child::auteur`
`.//auteur`	`descendant-or-self::node()/child::auteur`
`auteur[2]`	`child::auteur [position() = 2]`

XPath : syntaxe générale

La syntaxe abrégée

La notation **//** implique de **repartir depuis la racine** ; si l'on veut un **descendant-or-self** depuis le **noeud courant**, on écrira **.//**
Il faut prendre garde également au fait que **//auteur[2]** n'est pas équivalent à **/descendant-or-self::toto[position()=2]**
mais à

```
/descendant-or-self::node()/child::toto[position()=2]
```

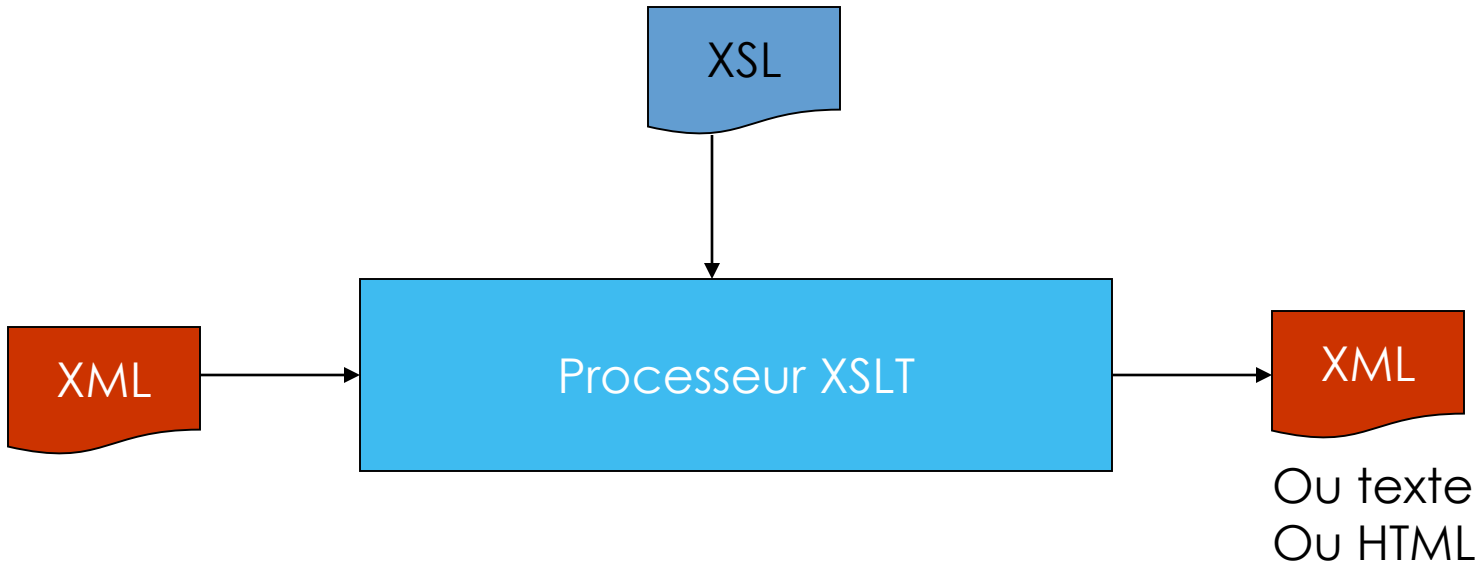
Autrement dit :
//auteur[2] fournit tous les nœuds auteur qui sont deuxième fils de leur père ;

```
/descendant-or-self::auteur[position()=2]
```

désigne **un unique nœud, le deuxième nœud** auteur du document.

XSLT

eXtensible Style Language Transformation



Traitement XSLT

- Effectué sur une liste de nœuds
  - Initialement cette liste ne contient que le nœud racine
- Recherche des templates correspondant aux nœuds de la liste.
- Exécution des templates
  - Ecriture sur la sortie
  - Mise à jour de la liste

Une feuille de style XSLT

- Un document XML
- Prologue Spécifique

```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/Transform">
```

- Des éléments caractérisant la feuille de style
  - xsl:import
  - xsl:include
  - xsl:output
  - …
- Des **Templates**

Les Templates XSLT

- Les templates XML définissent la transformation
- Identification d'une source
  - XPath
- Traitement (Ex : création d'élément XML)
  - Contenu du template
- Ex:

```
<xsl:template match="/">
        <H1> <xsl:apply-templates/> </H1>
</xsl:template>
```

Les Templates XSLT

```
<xsl:template
    match = pattern (XPath location)
    name = qname (pour Appeler un template)
    priority = number (règles d'ordre)
    mode = qname (Pls traitements / Elt>
    <!-- Contenu -->
</xsl:template>
```

Exemples de Template

- Une partie de document XML
  - This is an <emph> important </emph> point
- Un template d'une feuille de style

```
<xsl:template match="emph">
   <fo:inline-sequence font-weigth="bold">
      <xsl:apply-templates/>
   </for:inline-sequence>
<xsl:template>
```

- Le résultat
  - This is an **important** point

Contenu d'un template

- Si un template contient du texte, ce texte sera écrit sur la sortie.

Ex :

```
<xsl:template match="/">
   Feuille de style remplacant tout par ce texte ?!!!
</xsl:template>
```

Contenu d'un template

- Si un template contient des éléments XML ainsi que des attributs XML ceux ci seront écrit sur la sortie.

  Ex :

```
<xsl:template match="/">
<HTML>
    <HEAD>…
        </HEAD>
        …
</xsl:template>
```

- Il est néanmoins conseillé d'utiliser les fonctions de création d'élément XML

Fonctions de création

- Création d'un élément

  `<xsl:element name = qname>`
- Création d'un attribut

  `<xsl:attribute name = qname> valeur`
- Création de texte

  `<xsl:text > texte`
- Création d'instructions de traitement

  `<xsl:processing-instruction name=qname>`
- Création de commentaires

  `<xsl-comment> text`

Fonction xsl:apply-templates

- Fonction permettant d'appliquer un template sur les fils du nœud courant (met à jour la liste de nœuds)

```
<xsl:apply-templates
 select = expr
 mode = qname>
 contenu
</xsl:apply-templates>
```

Ex :

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

Fonction xsl:value-of

- La fonction xsl:value-of créer un nœud text correspondant à la valeur du résultat de l'expression

```
<xsl:value-of
select = string-expr >
```

Ex:

```
<xsl:template match="person">
   <p> <xsl:value-of select="@given-name"/> </p>
</xsl:template>
```

Répétition

- Il est possible d'appliquer un template à un ensemble d'éléments.

```
<xsl:for-each
 select = node-expr>
 Contenu
</xsl:for-each>
```

Ex:

```
<xsl:template match="/">
 <xsl:for-each select="eleve"> …
```

Traitement conditionnel

- XSLT propose deux instructions conditionnel

```
<xsl:if
   test = boolean-exp>
   Contenu
</xsl:if>
```

```
<xsl:choose>
   <xsl:when test=boolean-expr> … </xsl:when>
   <xsl:when test=boolean-expr> … </xsl:when>
   …
   <xsl:otherwise> … </xsl:otherwise>

</xsl:choose>
```

Et encore

- Possibilité de trier les éléments de la liste. Utilisé après un xsl:apply-templates ou un xsl:for-each

  `<xsl:sort select=`*`string-expr`* `…>`

- XSLT permet de créer des variables qui pourrons être utilisées dans la feuille de style. Une variable peut être un objet de n'importe quel type.

  `<xsl:varaible name=qname select=expr >`

- Autres fonctions XSLT
  - Opération sur les nombres
  - Opération sur les chaînes de caractères
  - Envoi de messages
  - Copy, …

Modes

- Les modes permettent à un élément d'être traité plusieurs fois, en produisant un résultat différent à chaque fois.
- <xsl:template match="fils" mode="m1">
  …
  <xsl:template match="fils" mode="m2">
  …
- <xsl:apply-templates mode="m2">

Template nommé

- L'utilisation de l'attribut **name** de l'élément **xsl:template** permet de spécifier un nom de modèle.
- Un template nommé peut ne pas avoir d'attribut **match**
- Un template nommé peut être appelé par <xsl:call-template name=*qname*>

Import de template

- XSLT founit deux mécanismes pour combiner des feuilles de style
  - <xsl:include href=*uri-reference*/>
    Copier/Coller de la feuille de style
  - <xsl:import href=*uri-reference*/>
    Liens vers la feuille de style importé. En cas de conflit de template, les feuilles importées sont moins prioritaire.

Feuille de style XSLT

- Transformer un arbre
- Feuille de style = ensemble de règles
- Association motif / modèles
- Objectif :
  - Production de texte
  - Personnalisation d'un document
  - Réorganisation
  - …

Feuille de style CSS

- Cascading Style Sheet
- Langage très simple (non XML)
- CSS1 : décembre 96 CSS2 : mars 98
- Encore mal supporté par les browsers (sauf Opera 4.02)
- Liste de règles qui décrit les éléments
- Pour chaque élément (Couleurs et image, Polices de caractère, Textes, Boites …etc)
  - Classification (style de listes, affichage par bloc…)

CSS dans XML (ou HMTL)

- But : présentation vers browser
- Via instruction de traitement

```
<?xml-stylesheet  type = "text/css" href="styleCD.css" ?>
```

- href désigne l'URI de la feuille

XSL-FO (Formatting Objects)

- Langage XML pour formater
- Finalisé par W3C fin Octobre 2001
- Similaire à HTML/CSS mais beaucoup plus puissant
- Outils permettant de générer PDF (FOP)
- Devrait plaire aux utilisateurs Latex…

Principe de XSL-FO

- Découpage par pages
- Notion de boites
  - Contenu (texte, espace, images, objets…)
  - Type (région, bloc, ligne, boite en-ligne,…)
  - Hiérarchie
  - Caractéristiques (position, fontes…)
- Pages maîtres
- Séquences de pages
- Numérotation
- …

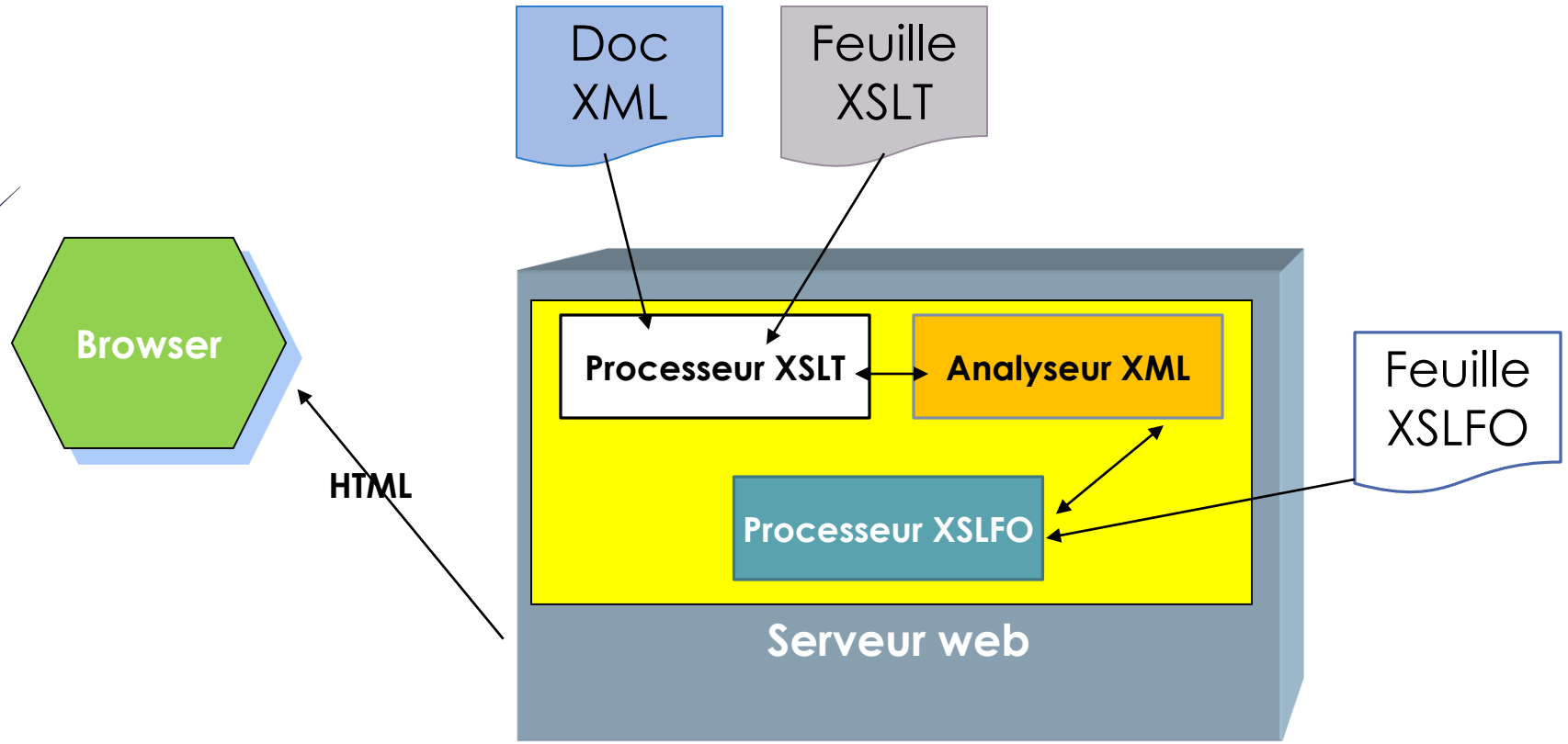
API SAX

- Application Programming Interface
- Mécanismes standardisés de manipulation de documents XML
- API événementielle
  - Processeur analysant le document
  - Association d'un gestionnaire de document
  - Activation des méthodes durant l'analyse
- Simple et efficace
- Implémentations JAVA (Xerces), C, C++

API DOM - Document Object Model

- SAX simple mais parfois limité (*modification de structure impossible*)
- Analyse puis création d'une structure
- Adaptable (mais plutôt orienté objet…)
- Spécification en IDL (OMG)
- Implémentations en Java, javascript

XML et le Web


Doc
XML
Feuille
XSLT
Browser
HTML
Processeur XSLT
Analyseur XML
Feuille
XSLFO
Processeur XSLFO
Serveur web

Interro

XML	eXtensible Markup Language
W3C	World Wide Web Consortium
DTD	Définition de Type de Document
XSD	XML Schema Document
XPATH	XML Path
XSLT	eXtensible Style Language Transformation
CSS	Cascading Style Sheet
URI	Universal Ressource Information
XSL-FO	eXtensible Style Language Formatting Objects
XLINK	XML Link
XPOINTER	XML Pointer
API	Application Programming Interface
DOM	Document Object Model
IDL	Interface Definition Language