W3C

XML

by *Thierry MICHEL*

Table of contents

Contenu du cours

1-XML un standard d'échange de données

- Qu'est ce que XML ?
- XML va t'il remplacer HTML ?
- Qui est responsable de XML ?
- Les navigateurs/outils XML

2- Créer des documents XML

- Les documents biens formes
- Les documents valides
- DTD et XML schema
- Parser XML
- Les bénéfices d'XML

3- La famille XML

- XLL: les liens hypertextes Xlink/Xpointer
- XSL: transformations de structures (XSL:T) et mise en forme (XSL:FO)
- CSS: styler des documents XML
- DOM le model objet document (Manipuler et naviguer XML)
- Les espaces de noms (Namespaces)
- Les Metadata RDF

4- Étude de cas

- Mise en oeuvre des technologies XML/DOM/XSL/XLL/CSS/HTML

5- Applications XML

- XHTML
- SMIL
- MathML
- RDF
- SVG

L'Histoire ...

- 1969: ARPANET financé par DoD
- 1974: TCP/IP (Transmission Control Protocol / Internet Protocol)
- 1977: Mail (SMTP)
- 1979: Usenet (NNTP)
- 1984: DNS (Domain Name Server)
- 1990: WWW développé par le CERN (Premier WWW serveur: code source gratuit).
- 1992: Premier GUI browsers Erwise, Viola.
- 1993: NCSA Mosaic.code source gratuit. Inclus les graphiques.
- 1994: Consortium W3C
- 1995: Netscape Browser 1.0
- 1998: XML1.0, SMIL10
- 1999: RDF, DOM1, XHTML10
- 2000: XLink, Xpointer, XHTML1.1, SVG, XML Schema, etc...

W3C: World Wide Web Consortium

Mener le Web a son plein potentiel

Mission: Consensus sur l'interopérabilité du Web.

- Développement des protocoles du web
- Menace de fragmentation
- Nouvelles fonctionnalités
- Un acteur neutre

Qu'est-ce que le W3C?

- Directeur: Tim Berners-Lee (Inventeur du Web)
- Sites: MIT/LCS / INRIA / Keio
- Équipe: ~60 de 13 nationalités
- Membres: ~500
- Offices: (UK, DE, IT, GR, MO, NL, SY, HK,TW, etc)

et un site web http://www.w3.org

Que fait le W3C?

Services aux membres:

- Organise des réunions de groupes de travail
- Facilite les discussions et l'obtention d'un consensus
- Fournit l'expertise nécessaire à la rédaction des spécifications techniques
- Promeut une architecture saine et extensible pour l'avenir

Services publics:

- Une base d'information sur le World Wide Web pour les développeurs et les utilisateurs
- Des programmes ouverts et gratuits mettant en œuvre ces spécifications et protocoles
- Des outils de validation (HTML, HTTP, CSS, WAI, ...)
- La possibilité de participer au processus d'élaboration des spécifications

Les spécifications en cours de développement au W3C sont rendues publiques régulièrement.

Processus du W3C

- **Groupes:** Groupes d'intérêt (IG), Groupes de travail (WG)
- **Documents:** Working Drafts, PWD, Candidate Rec.,Proposed Rec., Recommendations
- Soumissions (Submissions)

Domaines Techniques

Organisés en quatre Domaines:

1. Web Architecture [Arch]
2. User Interface [UI]
3. Technology and Society [T&S]
4. Web Accessibility Initiative [WAI]

Architecture [Arch]

- Hypertext Protocols (HTTP1.0, HTTP1.1)
- Extensible Markup Language: XML
- Document Object Model (DOM)
- Télévision et le Web
- Adressage (URL, URI, etc)
- Web Characterization
- Code: *Jigsaw - serveur HTTP/1.1 Java*

User Interface [UI]

- Hypertext (XHTML, HTML4.0)
- Présentation (CSS2, XSL)
- Graphiques: (PNG, SVG)
- Multimédia: SMIL
- Mathematiques: (MathML)
- Browser vocaux
- Accès pour Mobiles
- Internationalisation (I18N)
- Code: *Amaya - browseur/éditeur*

Technology and Society [T&S]

- Metadata/Sélection de Contenu (RDF, PICS1.1)
- Données Personnelles (P3P)
- Signatures Digitales (XML signatures)
- Commerce Electronique (MicroPayment)
- *Intellectual Property Rights (IPR)*

Web Accessibility Initiative [WAI]

- Revue des protocoles et langages spécifiés par le W3C
- Règles d'accessibilité au web: contenu, outils d'édition, et browseur
- Évaluation et correction
- Éducation

Architecture du World Wide Web

- URL: Universal Ressource Locator
- HTTP: 1.1 Hypertext Transfert Protocol
- HTML: 4.0 Hypertext Markup langage

URL : Universal Ressource Locator

Une adresse unique: hostname + Chemin+ nom de Fichier

```
http://www.w3.org/Consortium/Process/Overview.html
http://193.51.208.67/Consortium/Process/Overview.html
```

- **Correspondance IP adresses/hostnames**
- .com, .org, .net, .edu, .mil, .gov
- .fr, .uk, .jp, etc

HTTP 1.1 (Hypertext Transfert Protocol)

HTTP spécifie les messages échanges entre un serveur HTTP et un client.

les messages HTTP sont soit des **requêtes** du client vers le serveur soit des **réponses** du serveur vers le client.

- **HTTP request** : (GET, PUT, HEAD, POST, DELETE ...

```
GET http://www.w3.org/Overview.html HTTP/1.1
Accept-Language: fr, en-US, en-cockney, i-navajo
Accept-Charset: iso-8859-5, unicode-1-1
```

- **HTTP response:**

  - 200: Success - l'action est reçue, comprise et acceptée
  - 400: Client Error - la requête contient une mauvaise syntaxe
  - 500: Server Error - Le serveur n'a pu traiter une requête qui semble pourtant valide

- ```
  HTTP/1.1 200 OK
  Date: Wed, 22 Sep 1999 13:03:57 GMT
  Server: Apache/1.3.6 (Unix) PHP/3.0.11
  Last-Modified: Fri, 17 Sep 1999 14:30:14 GMT
  Content-Type: text/html; charset=iso-8859-1
  <HTML>
  ```

Demo Headers HTTP: http://www.w3.org/Overview.html,headers
  ```

HTML: des Documents structurés

- Structure de type documentaire pour publier des documents sur le web
  - Utilise un ensemble prédéfini de balises pour structurer les données
  - Multimédia (textes, images, sons, vidéo)
  - Global hypertexte
- Format non propriétaire base sur SGML (Standard Generalized Markup Language).
- Format textuel qui peut être créer par de simple éditeurs ou des éditeurs sophistiques WYSIWYG
- Séparer le fond et la forme
- Meilleure accessibilité

```
<HTML>
  <HEAD>
    <TITLE>Example de HTML</TITLE>
  </HEAD>
  <BODY>
      <H2>Example de lien</H2>
     <P> Visitez le  <A href="http://www.w3.org">W3C.</A></P>
  </BODY>
  </HTML>
```

Ce qui s'affiche:

Example de lien

Visitez le W3C.

HTML version 3.2 / 4.0

(HTML version 3.2)	(HTML version 4.0)
<P><FONT size="4" color="red"> stylé avec les attributs de FONT </FONT></P>	<P> le style est déprécié </P>

CSS: Cascading StyleSheets

- Séparer la structure d'un document et sa présentation
- Attacher des feuilles de style (stylesheets) a des document structurés
- Spécifier la présentation d'une famille de documents
- Décrire comment les documents sont visualisés a l'écran, imprimés, ou prononcés (browsers vocaux).
- Spécifier l'affichage en fonction des capacités des terminaux (taille de l'écran, couleur, résolution)
- Fonctionne avec HTML, XML, SVG, SMIL, XHTML, etc.

(HTML)	**(HTML + CSS)**

`<P>`
`<FONT size="4" color="red">`
`stylé avec les attributs de FONT`
`</FONT>`
`</P>`
	`<link href="monstyle.css" type="text/css">`
	
	`<P>stylé avec le fichier CSS</P>`
	--- 	
	`Fichier: monstyle.css`
	`P {font-size: 20px;  color: red;}`

Exemple Cascading StyleSheets

Démo CSS:

- Fichier sans CSS, avec un style sheet, autre style sheet
- Mise en page W3C

XML: Introduction

Qu'est ce que XML ?

- XML eXtensible Markup Language (*langage de balises extensible*)
- Métalangage pour creer des langages
- Basé sur SGML (Standard Generalized Markup Language, ISO 8879)
- Extensible (a la différence de HTML)
- Séparation du fond et de la forme

Qui est responsable de XML ?

- XML1.0 est une Recommandation du W3C
- XML est un format public et accessible a tous

XML remplace HTML?

- XML est complémentaire de HTML
- XML est plus rigide que HTML (syntaxe rigoureuse)
- XML est extensible (définir son propre jeu de balises)
- Liens plus riches
- Contrôle de validité

```
<HTML>
  <p>
  2004, route des Lucioles
  BP 93  06902 Sophia Cedex France
  </p>
```

```
<?xml version="1.0"?>
   <address>
       <number>2004</number>
       <street>route des Lucioles</street>
       <bp>BP 93</bp>
       <code>06902</code>
       <city>Sophia</city>
       <country>France</country>
   </address>
```

Avantages XML ?

- Documents **hypertextes et structurés**
- **Syntaxe stricte** (document bien formés)
- **Document Valides:**conformes a DTD (Document Type Definition)
- Format textuel facile a générer et a lire
- Format d'échange Universel
- Utilise des "balises" pour délimiter des data a lire par des applications

- Fichier peuvent être compressé (zip) ou a la volée HTTP1.1
- Recommandation du W3C et gratuit.

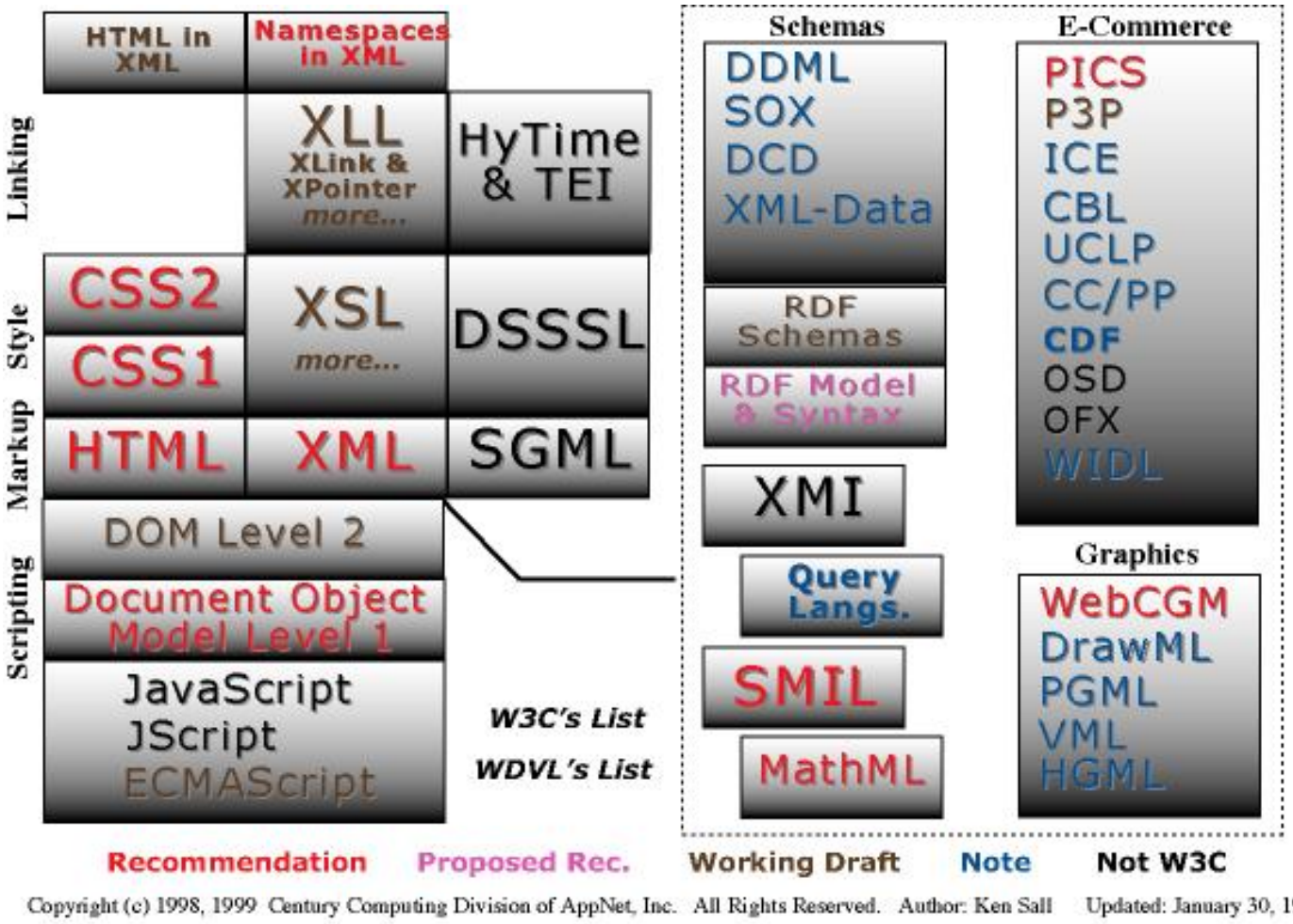
Les Navigateurs XML

- IE 5 et Netscape 6 ont un parseur XML

La Famille XML

- Styler des documents - (CSS2, XSL)
- Sélectionner des fragments de documents XML (XPath)
- Étendre les liens hypertextes (Xlink /Xpointer)
- Transformer des documents XML (XSLT)
- Fusionner plusieurs documents (XML-Namespaces)
- Spécifier les URl de Base (XMLBase)
- Représenter des graphiques vectoriels SVG
- Représenter des expressions mathématiques(MathML)
- Synchroniser des présentations Multimédia (SMIL)
- Modifier dynamiquement et naviguer dans des documents (DOM)
- Typer les données de formulaires (XForms)
- Extraire des informations: Query languages (XML-QL)

XML : permet d'automatiser des traitements de machines a machines.



XML: Documents bien Formés

- Nouveau concept introduit par XML
- Les document doivent être bien formés pour être correct
- Obéit a des règles syntaxiques strictes (Parser XML)
- Aucun vocabulaire prédéfini (sauf qques attributs xml:lang, xml:preserve)

Les règles Syntaxiques:

- **Tous les éléments doivent avoir des balises fermé**es
  CORRECT: Voici un paragraphe.
  INCORRECT: Voici un paragraphe
- **Tous les éléments doivent** respecter les caractères interdits
  CORRECT: <chiffre123> <espace_blanc>
  INCORRECT: <123chiffre> <espace blanc> <xmlbalise>
- **Tous les éléments doivent être correctement imbriqué**s
  CORRECT: Voici un paragraphe en italique.
  INCORRECT: Voici un paragraphe en italique
- **la casse des éléments est sensible**
  CORRECT: . un paragraphe.
  INCORRECT: un paragraphe.</P>
- **les éléments vides doivent avoir une fin de balise />**
  CORRECT:
<hr/>
  INCORRECT:
<hr>

  CORRECT:
  INCORRECT: <img src="karen.jpg" alt="Karen" >
- **les valeurs des attributs sont entre cotes ou double cotes**
  CORRECT: <table rows="3">
  INCORRECT: <table rows=3>
- **les valeurs des attributs doivent être spécifiées**
  CORRECT: <dl compact="compact">
  INCORRECT: <dl compact>
- **Un élément racine qui est unique**

CORRECT:	INCORRECT:
<repertoire> <carte> <-- description --> </carte> <carte> <-- description --> </carte> </repertoire>	<carte> <-- description --> </carte> <carte> <-- description --> </carte>

Syntaxe réservée - attributs (xml:lang, xml:space)

```
<p xml:lang="fr">Ce paragraphe est en francais</p>
<pre xml:space="preserve">
        Family name              First name         Age
        Jackson                     Joe              38
        Brewer                      Jack             45
</pre>
```

Structure d'un document XML bien formé

Un document XML **bien formé** est composé de:

1. Un Prologue
2. Un élément racine
3. Un arbre d'éléments et leurs attributs
4. Les commentaires

Structure d'un document XML (2)

---

1. Le Prologue

a. La déclaration XML

- Un document XML débute par:
  - la version: `version="1.0"`
  - le jeu de caractère: `encoding=` (UTF-8 ou UTF-16, etc)
  - la complétude: `standalone=`(yes, no)

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
```

b. Les instructions de traitement

- permettent de transmettre de l'information a une application pour un traitement donné

Syntaxe: <?Application instruction+?>

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

c. La déclaration de type de document (DTD)

- Elle n'est pas obligatoire

2. L'élément racine

- Un élément unique : la racine

3. L'arbre d'éléments

Les éléments

- la forme la plus connue du balisage
  `<element> <element-fils>.....</element-fils></element>`
- Un élément peut contenir:
  - d'autres éléments
  - des données (contenu)
  - des références a entités (prédéfinies `<` ou générales `&entités`)
  - sections littérales (CDATA)
  - être un élément vide (avec ou sans attributs)
  - les espaces, tab, rupture de lignes sont pris en compte (`xml:space="preserve"`)

```
<exemple id ="simple">
<section>Exemple de section littérale:
<![CDATA[ <auteurs>Dupond &amp;Co.</auteurs>]]>
</section>
<fin id ="vide"/>
</exemple>
```

Les attributs

- les attributs ont un nom et une valeur qui apparaissent après le nom de l'élément
  `<element attribut="valeur">`

Les références aux entités

- permet d'inclure le contenu de fichiers externes
  `&entite;`

4. Les commentaires

<!-- commentaires -->

Structure document XML (3)

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>

<!--   Processing instructions -->
<?xml-stylesheet type="text/xsl" href="style/example.xsl"?>

<!--   DTD externe au document XML -->
<!DOCTYPE dossier  SYSTEM "dossier.dtd">

<!--    début de l'instance  -->

<!--    élément racine  -->
<dossier chemin ="c:\documentsXML">

<!--   l'arbre d'éléments  -->

  <fichier nom="exemple.xml">
    ce fichier contient un exemple pour &titre;
  </fichier>

  <fichier nom="livre.xml">
     ce fichier decrit la structure du document
  </fichier>
</dossier>
```

Bonne utilisation des attributs

Choix de faire figurer l'information

- comme valeur d'un attribut
- comme contenu d'un élément

Remarques:

- Une valeur d'attribut n'est pas visualisée dans un navigateur (sauf XSL explicite)
- Une valeur d'attribut n'est pas visualisée dans un éditeur XML (pop menu)
- Une valeur d'attribut n'est pas structurée, (le contenu d'un élément peut être un arbre)
- Les attributs ne sont pas ordonnés: (ordre n'a aucune signification)
- L'indexation des documents se fait sur le contenu d'éléments particuliers (meta)

Exemple:

```
<?xml version ="1.0" ?>
   <address>
       <number>2004</number>
       <street>route des Lucioles</street>
       <bp>BP 93</bp>
       <code>06902</code>
       <city>Sophia</city>
       <country>France</country>
   </address>
```

```
<?xml version ="1.0" ?>
   <address
       number="2004"
       street="route des Lucioles"
       bp="BP 93"
       code="06902"
       city="Sophia"
       country="France"/>
```

Exercice: Document Bien Formé

La DTD

Déclaration de Type de Document (Document Type Description)

- Elle définit la grammaire d'une classe de document
- Elle permet de communiquer de la meta-information a un parseur
- Elle peut être **interne au document XML**, (définition locale)
  ```
  <!DOCTYPE nom [..définition de lDTD "dossier" ..]>
  ```
- Elle peut être **externe au document XML**
  (indentifiant URL)
  ```
  <!DOCTYPE dossier SYSTEM "dossier.dtd">
  ```
  (indentificateur PUBLIC)
  ```
  <!DOCTYPE rec-xml PUBLIC "-//W3C//Recommendation XML1.0//EN"
  "http://www.w3.org/TR/1998REC-xml-19980210.html">
  ```
- Elle peut être **externe et interne au document XML**
  ```
  <!DOCTYPE dossier  SYSTEM "externe.dtd"[
   ...définition de DTD "interne" ... ]>
  ```
  En cas de conflit, la déclaration interne locale est prioritaire par rapport la partie externe

EX:DTD externe au document XML

EX:DTD interne au document XML

EX:DTD interne et externe au document XML

Structure de la DTD

Quatre types de déclaration

1. Type d'éléments
2. Liste d'attributs
3. Entités (Générales ou Paramètres)
4. Notations

Structure de la DTD (2)

Quatre types de déclaration

1. Type d'éléments

Identifier les noms des éléments et le modèle de leur contenu

- nom de l'élément
- contenu `(EMPTY, ANY, contenu des éléments fils, #PCDATA)` chaîne de caractères parsée
  ? zéro ou un élément
  * zéro élément ou plus
  + un élément ou plus
  par défaut: un élément et un seul

`Syntaxe : <!ELEMENT NomElement (contenu)>`

Exemples:

```
<!ELEMENT p ANY >
<!ELEMENT p (#PCDATA|span)>
<!ELEMENT bar EMPTY >
<!ELEMENT livre
(preface?,avant-propos?,introduction,partie+,(titre,chapitre)+,conclusion)>
<!ELEMENT section (par |chap)*>
```

2. Liste d'attributs

Elles identifient les attributs de chaque éléments, et leur valeur par défaut

- un nom attribut
- un type: `CDATA, ID, IDREF, IDREFS, ENTITY, NMTOKEN`
- une valeur par défaut `:#REQUIRED (obligatoire), #IMPLIED (facultatif),#FIXED (fixé a une valeur)`.

`Syntaxe : <!ATTLIST NomElement NomAttribut type Default>`

Exemples:

```
<!ATTLIST chapitre identifiant ID #REQUIRED >

<!ATTLIST fichier
          alt CDATA #FIXED "chaine de caractères"
          facultatif CDATA #IMPLIED
          reference ID  #REQUIRED
          xml:lang NMTOKEN #FIXED "fr"
          date (ISO|EN-exp |FR-exp) #REQUIRED
          liste (ordonnee|numerotee |alphabetique) "ordonnee" >
```

3. Entités

Déclarer des blocs d'information au format XML (référencées dans le fichier XML ou la DTD)
Les documents XML ne sont pas restreints a un document unique:
- Multiples pièces appelées "Entités"

- **Entités Générales** (référencées dans le document)
  - Entité Internes/Entités Externes

  `Syntaxe :<!ENTITY nom-entité "valeur de l'entité">`
- **Entités Paramètres** (référencées uniquement dans la DTD)
  - Entités Internes/Entités Externes

  `Syntaxe : <!ENTITY %nom-entité "valeur de l'entité">`

Exemples:

- Entités Internes générales (référencées au sein d'un document XML)
  `<!ENTITY titre "construire une application XML">`
  sera référencée dans un document XML par `&titre;`
  XML prédéfinit des entités générales :
  - < produit le crochet gauche, <
  - > produit le crochet droit, >
  - & produit lesigne, &
  - ' produit l'apostrophe, '
  - " produit la double quote, "
- Entités Externes générales
  `<!ENTITY include SYSTEM "include.xml">` **(Bien formé)**
  sera référencée dans un document XML par `&include;`
- Entités Internes paramètres (référencées au sein d'une DTD)
  `<!ENTITY %developpement "INCLUDE">`
  sera référencée dans une DTD par `%developpement;`
- Entités Externes paramètres
  `<!ENTITY %nouveau SYSTEM "nouveau.txt">`
  sera référencée dans une DTD par `%nouveau;`

Ex: Entités Internes/externes Générales
Ex: Entités Internes Parametres
Ex: Entités Paramètres et générales

4. Notations

- Notations sont déclarées dans la DTD
- Types spécifiques de données externes binaires (non analysées par le parseur)
- Cette information est passée a l'application qui en fait sa propre utilisation.
- Les données (entités) non-XML sont stockées hors de l'instance

Exemple: *format de données Tiff et l'application qui permet de les traiter*

```
<!NOTATION TIFF SYSTEM "mon-application-tiff.app">

<!ELEMENT img EMPTY >
<!ATTLIST img externalentity ENTITY #REQUIRED >
<!ENTITY maphoto SYSTEM "face.tiff" NDATA TIFF >
```

dans l'instance:

```
<p><img externalentity="maphoto">Mon portrait</p>
```

XML: Documents Valides

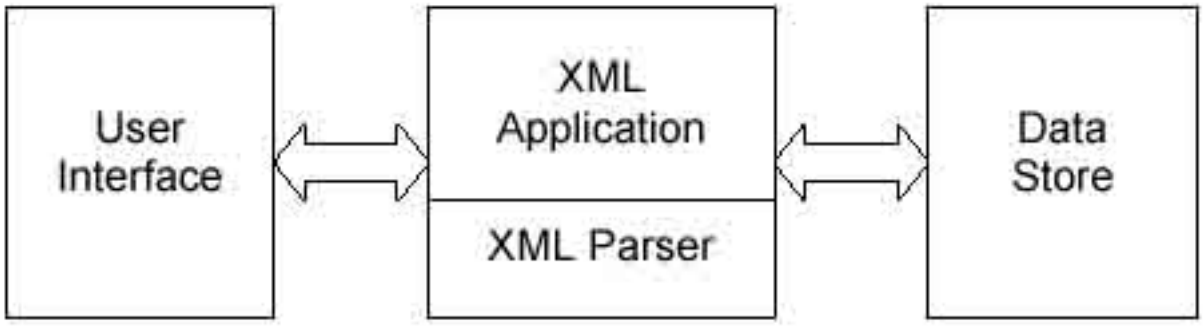
Un document valide est une instance de la structure définie par la DTD.

- Respecte la syntaxe XML (bien formé)
- Respecte la structure du document (prologue, DTD, elt racine, Arbre) (bien formé)
- Respecte la structure définie par la DTD
- Toutes les références sont résolues

Parseurs

Différents type de parseurs:

- Parseurs syntaxiques de document (bien formé)
- Parseurs validants (DTD ou Schema)
- Parsers compatibles avec Document Object Model (DOM)
- Parsers compatibles avec Simple API for XML (SAX)



Parseurs disponibles:

- **Java**
  -IBM’s parser, XML4J : www.alphaWorks.ibm.com/tech/xml4j
  - James Clark’s parser, XP : www.jclark.com/xml/xp
  - Sun’s XML parser : developer.java.sun.com/developer/products/xml/
  - Oracle's XML parser: technet.oracle.com/
  - DataChannel’s XJParser :xdev.datachannel.com/downloads/xjparser/
- C++
  - IBM’s XML4C parser : www.alphaWorks.ibm.com/tech/xml4c.
  - James Clark’s C++ : www.jclark.com/xml/expat.html.
  - Oracle's XML parser: technet.oracle.com/
- Perl

- Plusieurs parseurs XML en Perl: www.perlxml.com/faq/perl-xml-faq.html.

- Python
  - Plusieurs parseurs XML en Python: www.python.org/topics/xml/

Exemple de validation avec un parseur

- MSvalidator
- Visualisation d'arbre du fichier sample.xml
- Oracle parseur

Utilisation d'un parseur

```
//on charge le document XML source
   var source = Server.CreateObject("Microsoft.XMLDOM");
   source.load(Server.MapPath("document.xml"));
```

Exercice 1 : Écrire un document XML Valide
Exercice 2 : Écrire une DTD

Éditeurs XML

- éditeurs de texte simple (notepad)
- éditeurs de texte avec mode-macros XML (emacs)
- éditeurs XML (arbre du document et validant)

Le DOM

Document Object Model

- API - Interface de programmation pour les documents XML/HTML
- Définit la structure logique du document (arbre d'objets)
- Définit la façon dont une application peut y avoir accès et le manipuler (méthodes).
- Représente le document comme une hiérarchie de noeuds (Nodes)
- DOM level 1 est une Recommandation W3C, 1 Oct 1998
- DOM level 2 est une Recommandation W3C 13 Nov 2000: spécifie le DOM de CSS, la gestion d'évènements.

Indépendant du langage de programmation, et de la plate-forme (javascript, Jscript, Java, ECMAscript,C++)
Portabilité des applications XML d'un navigateur ou d'un serveur a un autre.

DOM: Les objets

La structure de DOM Core

- XML est une structure arborescente : Les noeuds de l'arbre contiennent les données
- DOM Core représente le document en une hiérarchie d'objets descendants de type "NODE"

Nom de l'objet	Description	Descendants possibles
Document	Racine de l'arbre	Element Racine, PI,Comment,DocType
Document fragment	Portion d'arbre	Element,PI,Comment,text,CDATA,Entityreference
Att	Attribut d'élément	Text, Entityreference
Element	Élément du document	Element, Text, comment, Entityreference
Text	Contenu textuel d'un element ou attribut	pas de descendant
Comment	commentaire	pas de descendant
CDATAsection	Texte contenant des données non parsees	pas de descendant
DocumentType	DTD	pas de descendant
Entity	Entité utilisée dans le document	Element, PI,Text, comment,CDATA,Entityreference
EntityReference	référence a une entité dans le document	Element, PI,Text, comment,CDATA,Entityreference
Processing Instruction		pas de descendant

W3C

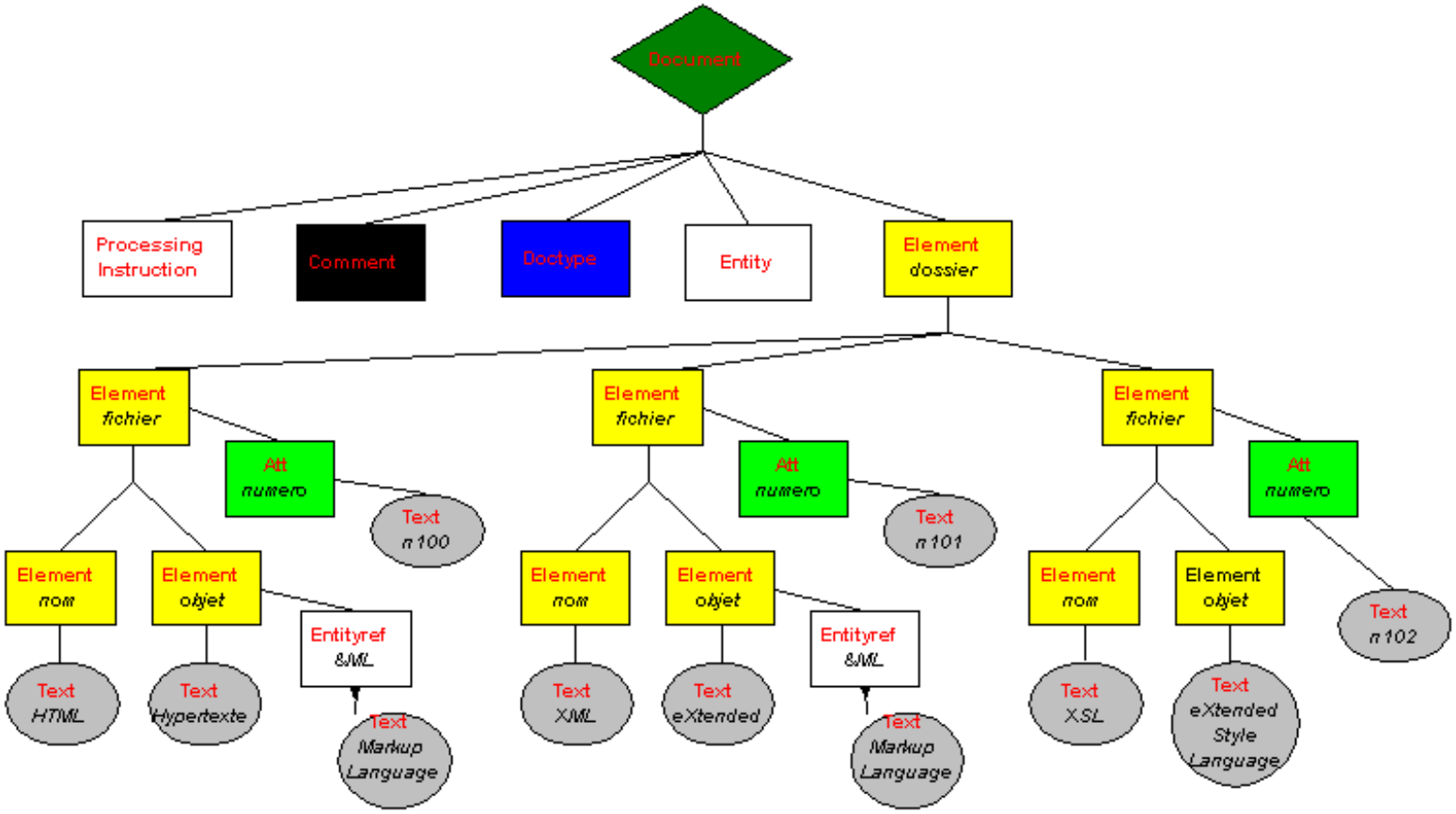
previ conte chang next

DOM: L'arbre d'objets

Le fichier XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href=dossier.xsl?>
<!-- DTD de dossier -->
<!DOCTYPE dossier [
<!ENTITY ML "Markup Language">
<!ELEMENT dossier (fichier)+>
<!ELEMENT fichier (nom, objet)>
<!ATTLIST fichier numero ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT objet (#PCDATA)>
]>
<dossier>
        <fichier numero="n100">
                <nom>HTML</nom>
                <objet>HyperTexte &ML;</objet>
        </fichier>
        <fichier numero="n101">
                <nom>XML</nom>
                <objet>eXtensible &ML;</objet>
        </fichier>
        <fichier numero="n102">
                <nom>XSL</nom>
                <objet>eXtensible Stylesheet language</objet>
        </fichier>
</dossier>
```

L'arbre d'Objets DOM correspondant


Document
Processing Instruction
Comment
Doctype
Entity
Element dossier
Element fichier
Att numero
Text n100
Element nom
Element objet
Entityref &ML
Text HTML
Text Hypertexte
Text Markup Language
Element fichier
Att numero
Text n101
Element nom
Element objet
Entityref &ML
Text XML
Text eXtended
Text Markup Language
Element fichier
Att numero
Text n102
Element nom
Element objet
Text XSL
Text eXtended Style Language

DOM: Les Interfaces

L'interface document

- Elle représente un document XML complet
- Ensemble des méthodes permettant de créer les objets du DOM nécessaires a un document

Methodes	Description
`CreateElement()`	`Création d'un élément`
`CreateDocumentFragment()`	`Création d'un fragment de document`
`CreateComment()`	`Création d'un commentaire`
`CreateAttribute()`	`Création d'un attribut`
.....etc.	

Accès au attributs d'un document:

Attributs	Description
`doctype`	`DTD associee au document (facultatif)`
`documentElement`	`Accès a l'élément Racine du document`

L'interface Element

Méthodes	Description
`getAttribute()`	`retourne la valeur d'un attribut a partir de son nom`
`removetAttribute()`	`supprime un attribut a partir de son nom`
`getElementByTagName ()`	`Retourne une collection ordonnée de tous les noeuds descendants du noeud courant`

.....etc.

Attribut:

Attributs	Description
tagName	Nom de l'élément (sa Balise)

L'interface Node

- interface qui définit le type de base de tous les noeuds
- Node modélise un noeud

Méthodes	Description
insertBefore()	Insère un noeud avant un noeud existant
replaceChild ()	Remplace un noeud par un autre et retourne le noeud remplace
removeChild ()	Enlève un noeud d'une liste de noeud et le retourne
appendChild ()	Ajoute un noeud a la fin d'une liste de noeud
.....etc.	

Accès au attributs d'un noeud:

Attributs	Description
nodeName	Nom du noeud (en fonction de son type: Att ou élément)
nodeValue	Valeur du noeud (en fonction de son type: Att ou élément)
FirstChild	Le premier fils du noeud
Attributes	Liste des attributs du noeud
.....etc.	

Exemple: Arbre d'objets DOM et Navigation

Démo Navigation dans un arbre de document avec DOM

Thierry MICHEL

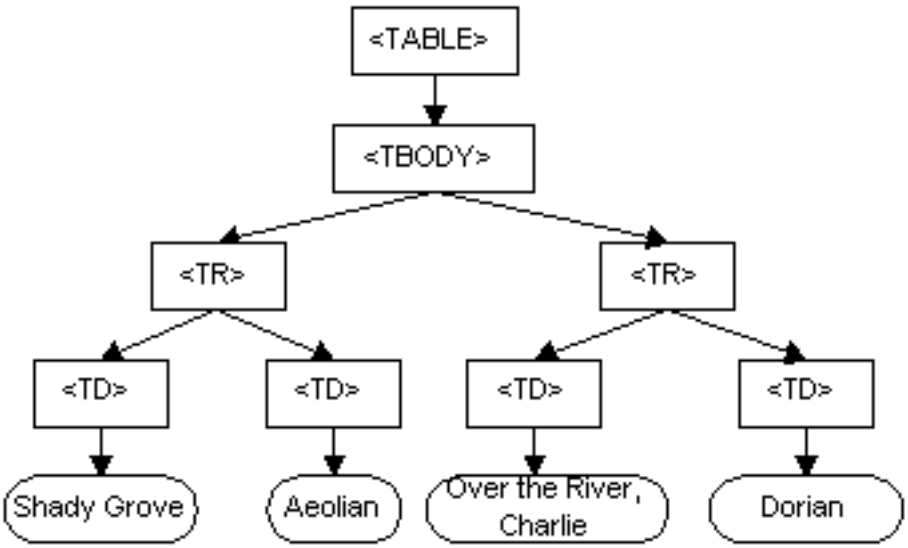
DOM HTML

La section "HTML" de DOM level 1 offre l'ensemble des objets et Interfaces permettant d'accéder a un document HTML4 et de le manipuler.

Par exemple considérons la ci-dessous table tirée d'un document HTML.

```
<TABLE>
<TBODY>
<TR>
<TD>Shady Grove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
<TD>Dorian</TD>
</TR>
</TBODY>
</TABLE>
```

La représentation DOM de cette table est :



W3C

previ conte chang next

DOM HTML: Les Interfaces

L'interface HTMLDocument

- Elle représente la racine d'un document HTML
- Ensemble des méthodes permettant de créer les objets du DOM nécessaires a un document

Methodes	Description
`open()`	`Ouvre un doc HTML pour écrire son contenu`
`close()`	`Ferme un doc HTML et l'affiche`
`getElementByid()`	`Retourne une collection d'éléments d'attribut "Name" donne`
.....etc.	

Les attributs d'un document:

Attributs	Description
`title`	`Titre du document <TITLE>`
`url`	`URL complete du document`
`body`	`Element qui contient le contenu du document <BODY> <FRAMESET>`
`images`	`Collection de toutes les images <IMG> du document`
`forms`	`Collection de tous les formulaires <FORM> du document`
`etc.`	

L'interface HTMLElement

- Interface de base de tous les éléments HTML du DOM

Les attributs :

Attributs	Description
`id`	identificateur de l'élément
`title`	titre de l'élément
`lang`	code de langue (RFC 1766)
`dir`	direction du texte et des tables
`className`	class de l'élément

Les éléments HTML

Liste des interfaces proposées par DOM level 1 et leurs équivalence HTML4.0

Interfaces	Tags
`HTMLHtmlElement`	`<HTML>`
`HTMLHeadElement`	`<HEAD>`
`HTMLTitleElement`	`<TITLE>`
`HTMLTableElement`	`<TABLE>`
`HTMLTableRowElement`	`<TR>`

Exemple d'utilisation de DOM HTML (création page HTML dynamique)

SAX:The Simple API for XML

Deux types d'APIs XML

1. **API basée sur un arbre (DOM)** : compile un document XML en un arbre qui permet a un application de naviguer dans cet arbre. Plus lourd mais plus performant (copie de fragments, utilisation répétée).
2. **API basée sur des évènements:**renvoi des évènements a l'application sans construire d'arbre. Plus simple, plus léger mais moins riche fonctionnellement.(extraction de qques éléments)

L'API SAX est une alternative pour travailler avec des documents XML.
C'est un standard *de facto* développé par David Megginson et les membre de XML-Dev .
Disponible www.megginson.com/SAX/.

Exemple:

```
<?xml version="1.0">
<doc>
<para>Hello, world!</para>
</doc>
```

l'API SAX renvoie une série d'évènements linéaires:

```
start document
start element: doc
start element: para
characters: Hello, world!
end element: para
end element: doc
end document
```

Une application utilise ces évènements comme des évènements d'une interface graphique utilisateur. Pas besoin de mettre en mémoire l'ensemble du document

Mais, il est également possible de construire un arbre en utilisant une API basée sur des évènements.

Parsers SAX

- IBM's XML for Java
- Sun's Java API for XML
- Oracle's XML Parser for Java
- etc..

Les Espaces de Nomage (Namespaces)

- une collection de noms, identifiés par une URL , utilisés dans les documents XML documents comme type d'élément ou d' attribut
- Permet de reconnaître dans un document, plusieurs vocabulaires
- Éviter les "collisions" lors de la fusion de documents utilisant les mêmes noms d'élément ou d'attribut.
- XML-Namespaces est une Recommandation W3C, 14 janv 1999

Préfixe qualifiant un nom d'élément:

```
<x xmlns:edi='http://ecommerce.org/schema'>
   <!-- l'element 'prix' est dans le  namespace is http://ecommerce.org/schema -->
   <edi:prix units='Euro'>32.18</edi:prix>
</x>
```

Préfixe qualifiant un nom d'attribut

```
 <x xmlns:edi='http://ecommerce.org/schema'>
  <!-- le namespace de l'attribut 'taxClass' est http://ecommerce.org/schema -->
  <Article edi:taxClass="tva">lecteur CD</Article>
</x>
```

Espaces de nomage par défaut

La déclaration du namespace s'applique a l'élément ou il est déclaré et a tout ses éléments fils (sauf si un autre namespace est déclaré).

Syntaxe générale:

```
<?xml version='1.0'?>
 <fichier xmlns='http://www.w3.org/foo/schema'>
    <info>
       <--  le namespace s'applique a tout l'arbre -->
    </info>
</fichier>
```

Unicité des attributs

Dans un document XML aucun élément ne peut contenir deux attributs qui ont:

1. des noms identiques
2. des préfixes qui sont lies a un namespace identique.

Exemple, les éléments "incorrect" sont ici illégaux:

```
<!-- http://www.w3.org est lie a n1 et n2 -->
<x xmlns:n1="http://www.w3.org"
   xmlns:n2="http://www.w3.org" >
  <incorrect a="1"     a="2" />
  <incorrect n1:a="1"  n2:a="2" />
</x>
```

Les exemples ci dessous sont valides car le namespace par défaut ne s'applique pas aux attributs:

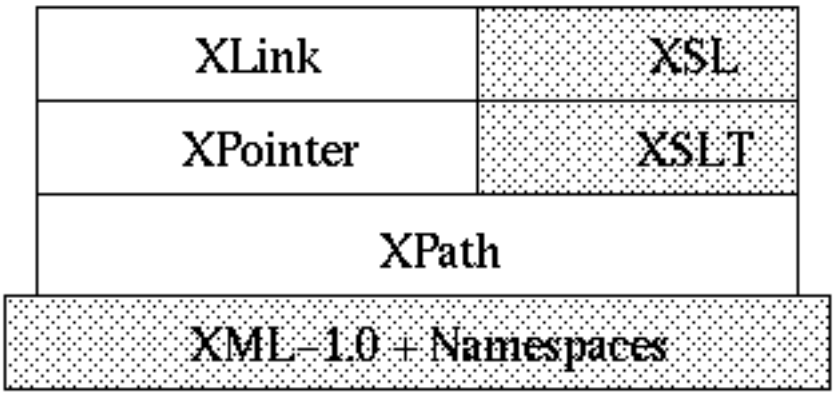
```
<!-- http://www.w3.org est lié a n1 et est le namespace par défaut -->
<x xmlns:n1="http://www.w3.org"
   xmlns="http://www.w3.org" >
```

```
  <correct a="1"     b="2" />
  <correct a="1"     n1:a="2" />
</x>
```

Exemple Déclaration d'espace de Nomages

- **Exemple d'espaces de nomage concernant un élément**
- **Example d'espaces de nomage par défaut**
- **Exemple de multi-espaces de nomage**

XML: XPath-Xpointer-Xlink-XSL



XML Path Language (XPath)

Qu'est ce que XPath ?

- permet de sélectioner des parties d'un document XML
- Xpath W3C Recommendation 16 Nov 1999
- XPath utilise une syntaxe compacte a utiliser dans les URI XPointer et des valeurs d'attributs XML XSLT
- XPath modélise un document XML en un arbre de noeuds (éléments, attributs, text).
- XPath permet de manipuler des chaines de caractères, des nombres et des booléens

Localisation XPath

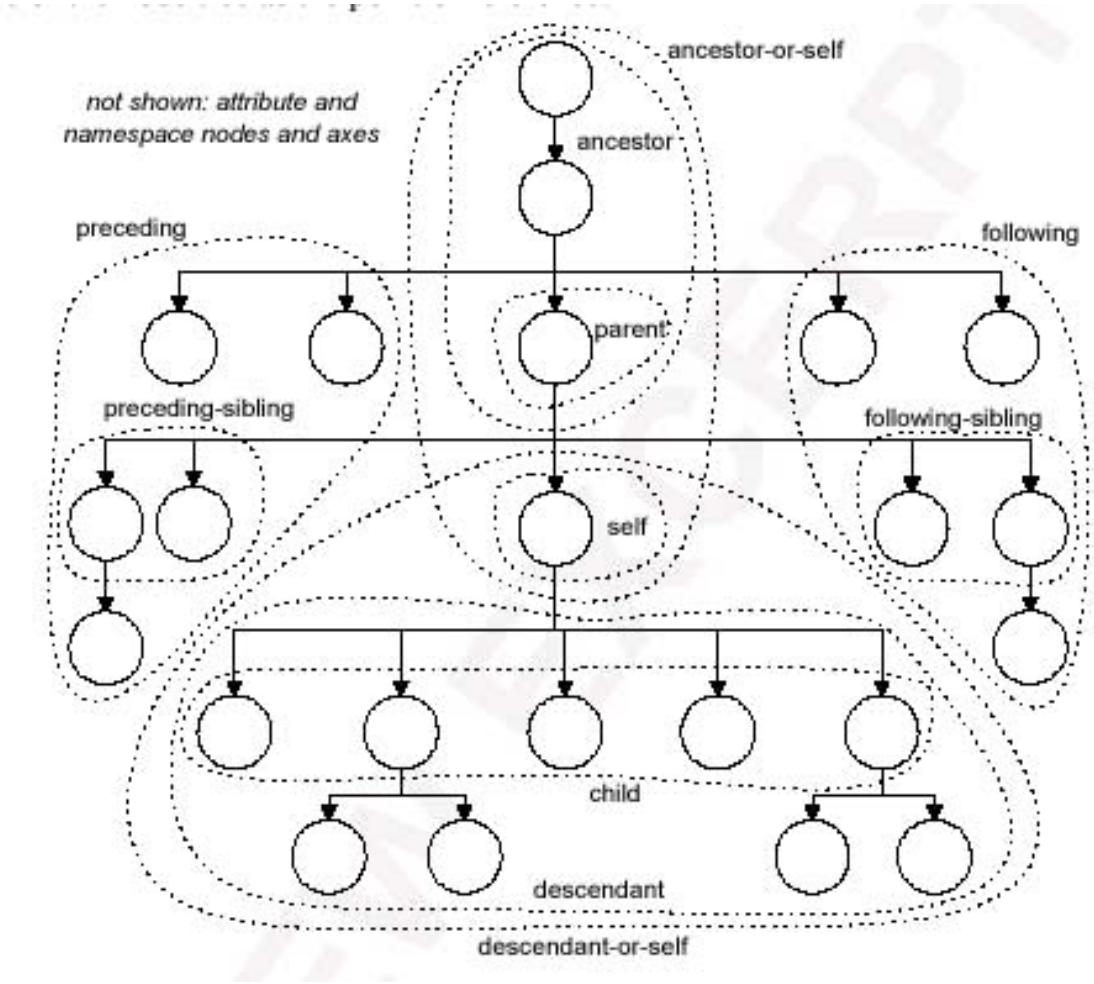
Trois parties:

1. **Un axe**, qui spécifie la relation entre les noeuds sélectionnés
2. **Node test**, qui spécifie le type de noeud
3. **Expressions** (zéro ou plusieurs) permettant de rafiner la collection de noeuds sélectionnée.

***Syntaxe* : Axe :: Node test [Exp1 Exp2]**

Exemple: `child::objet[position()=1]`,
`child` est le nom de l'axe, **`objet`** est le node test et **`[position()=1]`** est l'expression.

XPath- Les Axes



'self'
'ancestor'
'ancestor-or-self'
'attribute'
'child'
'descendant'
'descendant-or-self'
'following'
'following-sibling'
'namespace'
'parent'
'preceding'
'preceding-sibling'

XPath- Node Tests

Un node test peut être n'importe quel noeud.
Node test: `text() , comment(), processing-instruction()`
Exemples: `child::text()` *sélectionne le noeud text fils du noeud contextuel*

Syntaxe simplifiée

- **`child::` peut être omis (axe par défault).**
  `ex:`
  `fichier` *est le raccourci de* `child::fichier`. *Sélectionne tous les élements fils* `fichier`
  `fichier/objet` *est le raccourci de* `child::fichier/child::objet`. *Sélectionne tous les élements* `objet` *qui sont fils de l'élement fils* `fichier`
- **/ le chemin absolu vers le noeud racine du document**
  `/dossier` *Sélectionne l'élement racine* `dossier` *du document*
  `/dossier/fichier` *Sélectionne tous les élements* `fichier` *qui sont fils de l'élement racine* `dossier`
- **`//` est le raccourci de `/descendant-or-self::node()/`**
  `ex: //nom` *est le raccourci de* `/descendant-or-self::node()/child::nom`
  (Sélectionne tous les élements `nom` *du document)*
  `ex: dossier//nom` *est le raccourci de*
  `dossier/descendant-or-self::node()/child::nom`
  (Sélectionne tous les élements `nom` *qui sont descendants de l'élement* `dossier` *)*
- **`.` est le raccourci de `self::node()`.**
  `ex: .//fichier` *est le raccourci de*
  `self::node()/descendant-or-self::node()/child::fichier`
  (sélectionne n'importe quel élement `fichier` *descendant du noeud contextuel)*
- **`..` est le raccourci de `parent::node()`.**
  `ex: ../fichier` *est le raccourci de* `parent::node()/child::fichier`
  (sélectionne les fils `fichier` *des parents du noeud contextuel)*
- *** Sélectionne n'importe quel élement**
  `ex: //*` *Sélectionne tous les élements*
  `fichier/nom/*` *Sélectionne les élements fils de* `fichier/nom`
  `/*/*/objet` *Sélectionne les élements* `objet` *qui ont trois 2 ancètres*
- **attribute:: peut être abrévié en @**
  ex: `//fichier[@id]` *Sélectionne l'élement* `fichier` *qui a un attribute* `id`
  `//fichier[@*]`*Sélectionne l'élement* `fichier` *ayant un attribut quelconque*

Xpath- Expressions

Permet de filtrer une collection de noeuds en fonction d'une axe donné

Exemples:
`fichier[3]` *équivaut a* `child::fichier[position()=3]`
Sélectionne le troisième noeud fils `fichier` *du noeud contextuel*

`fichier[@numero="n102"]` *est le raccourci de*
`child::fichier[attribute::numero="n102"]`
sélectionne l'élément fils `fichier` *ayant un attribut* `numero` *dont la valeur est égale a* `n102`

Autres Expressions Xpath

Booléens : `or`, `and`, `=`, `!=`, `<=`, `<`, `>=`

Exemples d'expressions:
`//objet | //nom` *Sélectionne tous les élements* `objet` *ou* `nom`
`//individu[age='10']` *Sélectionne les éléments* `individus` *dont l'élément* `age` *est "10"*

`//fichier[system="Fat32"]/nom[last()]` *Sélectionne le dernier élement* `nom` *dans toutes les* `fichiers` *qui ont un élément* `system` *égal a "Fat32".*

Nombres

`//fichier[position() mod 2 = 0 ]` *Sélectionne les éléments pairs* `fichier`
`//section[Prix = 2*quantité]` *Sélectionne les* `sections` *ou le* `prix` *égal 2 * quantité.*

XPath: Exemples de location paths (Syntaxe non simplifiée)
XPath: Exemples de location paths (Syntaxe simplifiée)

Les Fonctions Xpath

1- Fonctions Node Set

- **Function:** *number* **last**() renvoie un nombre égal a la taille de la collection de noeuds `/AAA/BBB[last()]` *Sélectionne le dernier élément `BBB` fils de l'élement `AAA`*
- **Function:** *number* **position**() renvoie un nombre égal a la position contextuelle `para[position()=4]` *Sélectionne le 4 élément fils `para`*
- **Function:** *number* **count**(*node-set*) renvoie le nombre de noeuds de l'argument
- **Function:** *node-set* **id**(*object*) sélectionne les éléments d'après leur ID unique `id("foo")` *sélectionne l'élément ayant l'identifiant `foo`* `id("foo")/para[position()=5]` *sélectionne le cinquième `para` fils de l'élément avec l'unique ID `foo`*
- **Function:** *string* **local-name**(*node-set*?) renvoie la partie locale du nom étendu noeud
- **Function:** *string* **namespace-uri**(*node-set*?) renvoie le namespace URI du nom étendu noeud

2- Fonctions String

- **Function:** *string* **string**(*object*?) convertit un objet en une chaine comme suit:
  - Une collection de noeuds est convertie en chaine en renvoyant la "valeur de chaine" du premier noeud de la collection Si la collection est vide, une chaine vide est renvoyée.
  - Un nombre est converti en chaine.
- **Function:** *number* **string-length**(*string*?) renvoie le nombre de caractères de la chaine
- **Function:** *string* **concat**(*string*, *string*, *string**) renvoie la concatenation des arguments
- **Function:** *boolean* **starts-with**(*string*, *string*) renvoie "true" si le premier argument commence par la deuxième chaine argument, sinon renvoie "false"
- **Function:** *boolean* **contains**(*string*, *string*) renvoie "true" si le premier argument contient la deuxième chaine argument, sinon renvoie "false"
- **Function:** *string* **substring-before**(*string*, *string*) renvoie la substring du premier argument qui précède la première occurence du deuxième argument dans le premier.
  Ex: `substring-before("1999/04/01","/")` *renvoie `1999`.*
- **Function:** *string* **substring-after**(*string*, *string*) renvoie la substring du premier argument qui suit la première occurence du deuxième argument dans le premier.
  Ex: `.substring-after("1999/04/01","/")` *renvoie `04/01`*
  `substring-after("1999/04/01","19")` *renvoie `99/04/01`.*
- **Function:** *string* **substring**(*string*, *number*, *number*?) renvoie la substring du premier argument commencant a la position spécifie dans le deuxième arg avec la longueur spécifiée dans le troisieme arg.

Ex: `substring("12345",2,3)` *renvoie "234"*.
Ex:, `substring("12345",2)`*renvoie "2345"*.

- **Function:** *string* **normalize-space**(*string*?) renvoie la chaine sans espaces avant ni après. Les séquences de blancs sont remplacées par un seul blanc.
- **Function:** *string* **translate**(*string*, *string*, *string*) renvoie le premier arg. chaine avec les occurences de caractères du deuxième arg. remplacés par celui du troisième
  Ex: `translate("bar","abc","ABC")` *renvoie la chaine BAr*.
  Ex: `translate("--aaa--","abc-","ABC")` *renvoie "AAA"*.

3- Fonctions nombres

- **Function:** *number* **number**(*object*?) convertit ces arguments en un nombre comme suit:
  - une chaine qui comporte des espaces blancs ou signe moins est converti en un nombre sinon en NaN
  - booléen true est converti en 1; boolean false est converti en 0
  - a node-set est d'abord converti en une string puis en un nombre
- **Function:** *number* **sum**(*node-set*) renvoie la somme de chaque noeud de la collection
- **Function:** *number* **floor**(*number*) renvoie le plus grand entier inférieur a l'argument
- **Function:** *number* **ceiling**(*number*) renvoie le plus petit entier inferieur a l'argument
- **Function:** *number* **round**(*number*) renvoie l'entier le plus proche de l'argument

Exemple: `//BBB[position() = floor(last() div 2+0.5) or position() = ceiling(last() div 2+0.5)]`
Séléctionne le(s) élement(s) médiants BBB

Xlink

- Décrire des liens entre objets dans des ressources XML
- Les liens simples (Simple links) ont les mêmes fonctionnalités que les liens HTML <A>
- Les liens étendus (Extended links): liens a destination multiples.
- A namespace (préfixe avec "xlink") `http://wwww.w3.org/1999/xlink`
- Xlink est une W3C Candidate Recommendation

Les Propriétés (attributs):

`xlink:type`	attribut indique le type de lien - valeur `simple` lien simple (simple link) - valeur `extended` lien étendu (extended link)
`xlink:href`	l'attribut de localisation, sa valeur est une URI-Reference [RFC 2396].
`xlink:title`	attribut qui titre le lien
`xlink:role`	URL permettant de caractériser la fonction du lien
`xlink:show`	`"new", "replace", "embed".`
`(<A HREF="http://www.an.org" target="_blank">...</A>, target="_self")`	
`xlink:actuate`	évènement qui déclenche le lien. `"onLoad", "onRequest"`
`<IMG SRC="http://www.an.org/smiley.gif" ALT=":-)"> <A HREF="http://www.an.org" >...</A>`	
`xlink:type="locator"`	éléments contenus dans les liens étendus définissant une URL externe - `xlink:label`, permet de labeliser le locator - `xlink:title`, le titre du locator - `xlink:role`, URL permetant de caractériser la fonction du locator
`xlink:type="ressource"`	éléments contenus dans les liens étendus, définissant une URL locale - `xlink:label, xlink:title, xlink:role`, etc
`xlink:type="arc"`	éléments contenus dans les liens étendus définissant le comportement d'un lien avec les attributs suivants: - `xlink:from`, la source de l' arc - `xlink:to`, la destination de l'arc - `xlink:arcRole`, URL permetant de caractériser la fonction de l'arc - `xlink:title`, le titre de l'arc

Parent type	types des descendants
`simple`	aucun
`extended`	`locator`, `arc`, `resource`, `title`
`locator`	`title`
`arc`	`title`
`resource`	aucun
`title`	aucun

- **Un exemple de lien simple Xlink:**
- **Un exemple de lien multiple Xlink:**
- **Autre exemple de lien multiple Xlink, utilisant les arcs**

Xpointer

- Langage permettant de pointer dans la structure interne d'un document XML
- Permet de référencer des éléments, et autres parties de documents XML, (sans attribut ID, ni ancre).
- XPointer opère sur l'arbre d'éléments du document XML
- XPointer repose sur le langage XPath, et ajoute des fonctionnalités supplémentaires
- Xpointer est une Candidate Recommendation du W3C

Xpointer - Syntaxe

Xpointer = Renvoi absolu + Renvoi relatif

Renvoi absolu

1. **Root**: renvoi a la racine d'un document
   `http://www.w3.org/docs/mydoc.xml#xpointer(/)`
2. **ID**: renvoi a l'élément contenant cet identifiant
   `http://www.w3.org/docs/mydoc.xml#xpointer(id("identifiant"))`

Différentes syntaxes:

Full Xpointer
`http://www.w3.org/docs/mydoc.xml#xpointer(id("n102"))`
`http://www.w3.org/docs/mydoc.xml#xpointer(//*[@id="n102"])`

Bare Names (syntaxe simplifiée)
`http://www.w3.org/docs/mydoc.xml#n102` (Bare Names)

Child sequences

`#xpointer(id("n102")/*[2])`équivaut a `n102/2` *sélectionne le 2éme fils de l'élément dont ID est "n102".*

`http://www.w3.org/docs/pa.xml#xpointer(/fichier[3])`
renvoi au troisième élément "fichier" rencontré a partir de la racine du document

Namespaces

Déclaration `xmlns:ex="http://example.com/foo"`
`xpointer(//ex:fichier)`

Xpointer :Extentions à Xpath

Point - Séléctionne une position dans une fichier XML
`xpointer(start-point(//fichier))` *Positionne au début de l'émént* `fichier`
`xpointer(end-point(//fichier))` *Positionne a la fin de l'émént* `fichier`

Range - Séléctionne un intervalle dans une fichier XML
`xpointer(range(/))` *Sélectione le contenu du document*
`xpointer(range(//fichier/objet[1]))`*Sélectione la 1eme occurrence de l'élément* `objet` *et son contenu , fils de tout* `fichier`
`xpointer(range-inside(//fichier/objet[1]))` *Sélectione uniquement le contenu de la 1eme occurrence de l'élément* `objet` *fils de n'importe quel* `fichier`

Fonctions:

string-range () : Séléctionne un intervalle d'une chaine de caractères
Exemples:

`xpointer(string-range(//title,"Thierry Michel"[17]))`
Sélectione la 17th occurrence de la chaine "Thierry Michel" présente dans tout élément `title`

`xpointer(string-range(//*,'Thierry', 1, 3))`
Sélectione les 3 premiers caractères de la chaine "Thierry" présente dans tout élément

`xpointer(end-point(string-range(/title,'Thierry')))` *Positionne après la chaine de caractère Thierry de l'élément* `title`

range-to ()
Exemple1: `xpointer(id("chap1")/range-to(id("chap2")))`
Séléctionne l'intervalle depuis l'élement dont ID est "chap1" jusqu'a l'élement dont ID est "chap2".

Exemple dans un lien Xlink:

```
<lien_simple
xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple"
xlink:href="mydocument.xml#xpointer(/fichier[position() &lt; 2)])"
/>
```

Exemples Xpointer:

Identificateurs de ressources

```
<------------------------------ URI ------------------------------------------>

                                          <------- fragment identifier -------->
                                                   <--- expression Xpointer -->
http://www.foo.com/docs/mydoc.xml#xpointer#xpointer(article/section[position()=5])
```

Identificateurs absolus

- Host + `Chemin + fichier` `http://www.foo.org/docs/globe.jpg`
- Host + `Chemin + fichier +param`
  `http://www.foo.org/cgi/rech?q="paris+restaurants"`
- Host + `Chemin + # + string :` `http://www.foo.org/docs/pa.html#n101`
- Host + `Chemin + # + Xpointer:` `http://www.foo.org/docs/pa.xml#xpointer(/*[1])`
- Host + `Chemin + | + Xpointer:`
  `http://www.foo.org/docs/pa.xml|xpointer(id("n102")/*[3])`

Identificateurs relatifs

Si URL est absente, la ressource cible est le document source

- `xpointer(//*[@id="n102"])`
- `#xpointer(id("n102"))`
- `#n102`

XML Base

XMLBase permet:

- de définir une URL de base pour un document XML ou partie d'un document XML
- de résoudre des URL relatives (images externes, style sheets, applets, etc.)
- Similaire a HTML BASE
- XMLBase est une Candidate Recommendation du W3C
- La syntaxe consiste en un attribut XML: `xml:base`.

Règles de résolution des URL Relatives

- Une URL relative est résolue avec l'URL Base décrite par l'attribut `xml:base` du plus proche élément ancêtre ayant un attribut `xml:base`
- Une URL relative d'un attribut est résolue avec l'URL Base décrite par l'attribut `xml:base` de l'élément qui porte cet attribut s'il existe, sinon par l'attribut `xml:base` du plus proche élément ancêtre ayant un attribut `xml:base`

Exemple de `xml:base` dans un document XHTML simple.

```
<?xml version="1.0"?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xml:base="http://example.org/today/">
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <p>See <a href="new.xml">what's new</a>!</p>
    <p>Check out the hot picks of the day!</p>
    <ol xml:base="/hotpicks/">
      <li><a href="pick1.xml">Hot Pick #1</a></li>
      <li><a href="pick2.xml">Hot Pick #2</a></li>
      <li><a href="pick3.xml">Hot Pick #3</a></li>
    </ol>
  </body>
</html>
```

Les URL dans cet exemple sont resolues comme suit:

- "what's new" est résolu en l'URL "http://example.org/today/new.xml"

- "Hot Pick #1" est résolu en l'URL "http://example.org/hotpicks/pick1.xml"
- "Hot Pick #2" est resolu en l'URL "http://example.org/hotpicks/pick2.xml"
- "Hot Pick #3" est resolu en l'URL "http://example.org/hotpicks/pick3.xml"

Afficher XML

Deux méthodes pour afficher le XML

- XSL: eXtended Stylesheet Language
- CSS: Cascading StyleSheet

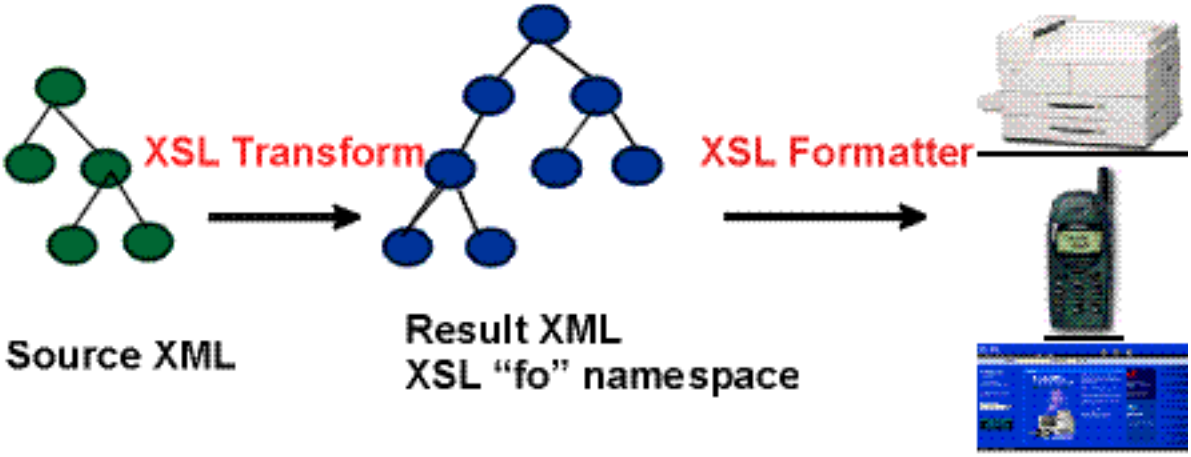
CSS: Cascading Style Sheets

- Un langage simple permettant de styler les documents Web (fonts, couleurs, positionnement)
- Facile a apprendre, utiliser et a maintenir
- Nombreux éditeurs de CSS en WYSIWYG
- CSS1 et CSS2 sont des Recommandations du W3C

XSL: Extensible Style Langage

- Un autre langage pour appliquer des feuille de style: plus puissant, plus complexe et spécifique a XML
- XSL se compose deux parties:
  1. **Un langage de transformation de documents XML (XSL-T)**
     - Transformer la structure d'un document (produire des tables de matières, index, etc.)
     - Transformer des données XML en documents HTML/CSS (sur le Web server)
     XSLT est une Recommandation du W3C du 16 Nov 1999
  2. **Un langage de formatage : XSL-FO (XSL Formatting Objects);**
     - Les propriétés de CSS et plus.
     XSLFO est une Candidate Recommandation du W3C, 21 Nov 2000

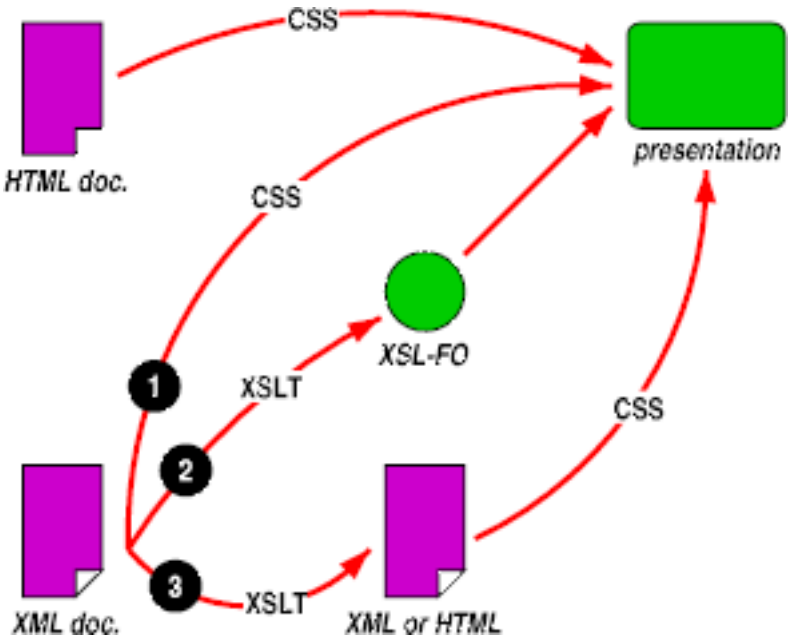
XSL Two Processes: Transformation & Formatting



Result XML is the result of XSLT processing. Syntactically, it is XML elements and attributes.

XSL ou CSS ?

Les documents XML peuvent être affiches de trois façons:



(1) si le document n'a pas a être transforme utiliser CSS.
Sinon, utiliser XSL-T, le langage de transformation XSL, d'une des deux façons:
- (2) générer les propriétés du style avec XSL-FO (XSL Formatting Objects);
- (3) générer un nouveau document XML ou HTML associe a une CSS style sheet pour ce nouveau document.

	CSS	**XSL**
Peut être utilisé avec HTML	Oui	Non
Peut être utilisé avec XML	Oui	Oui
Est un langage de Transformation	Non	Oui
Syntaxe	CSS	XML

Associer un style a un document XML

Les Style Sheets sont associées a un document XML grâce a une "Processing Instruction" dont la cible est "xml-stylesheet"

CSS Style Sheets

HTML:

```
<HEAD>
<LINK href="mystyle.css" rel="style sheet" type="text/css">
```

XML:

```
<?xml version='1.0'?>
<?xml-stylesheet href="mystyle.css" type="text/css"?>
```

XSL Style Sheets

Exécuté par le client

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

Exécuté par le serveur
(Exemple sur un serveur IIS)

Démo : Exemple fichier Dossier XML/CSS - Dossier XML - Dossier css

Qu'est ce que XSLT?

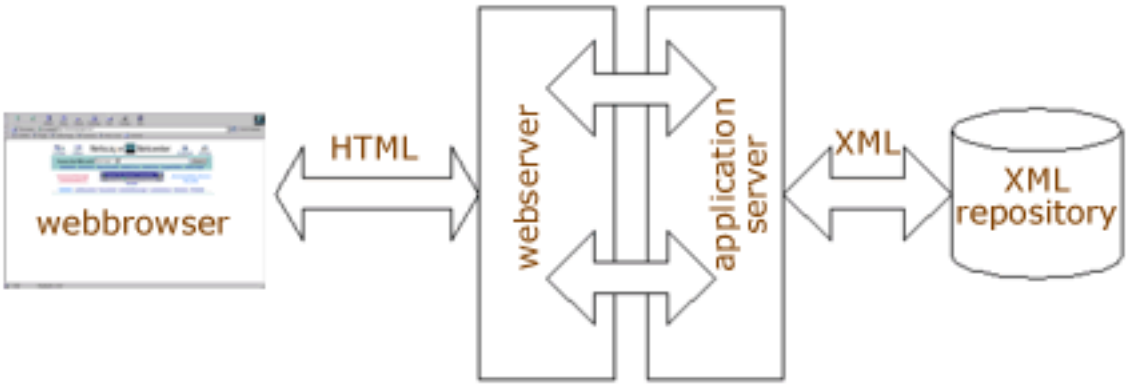
- Une grammaire XML (DTD)
- Transforme un arbre XML en un autre arbre XML (ou du HTML)
- Déclaratif, pas de code requis
- Une méthode intégrée au DOM XML (transformNode)

Règles de transformation:

- Des règles implicites appliquées aux éléments XML en absence de règles explicites
- Ces règles permettent des transformations récursives sur les éléments qui n'ont pas de règles explicites

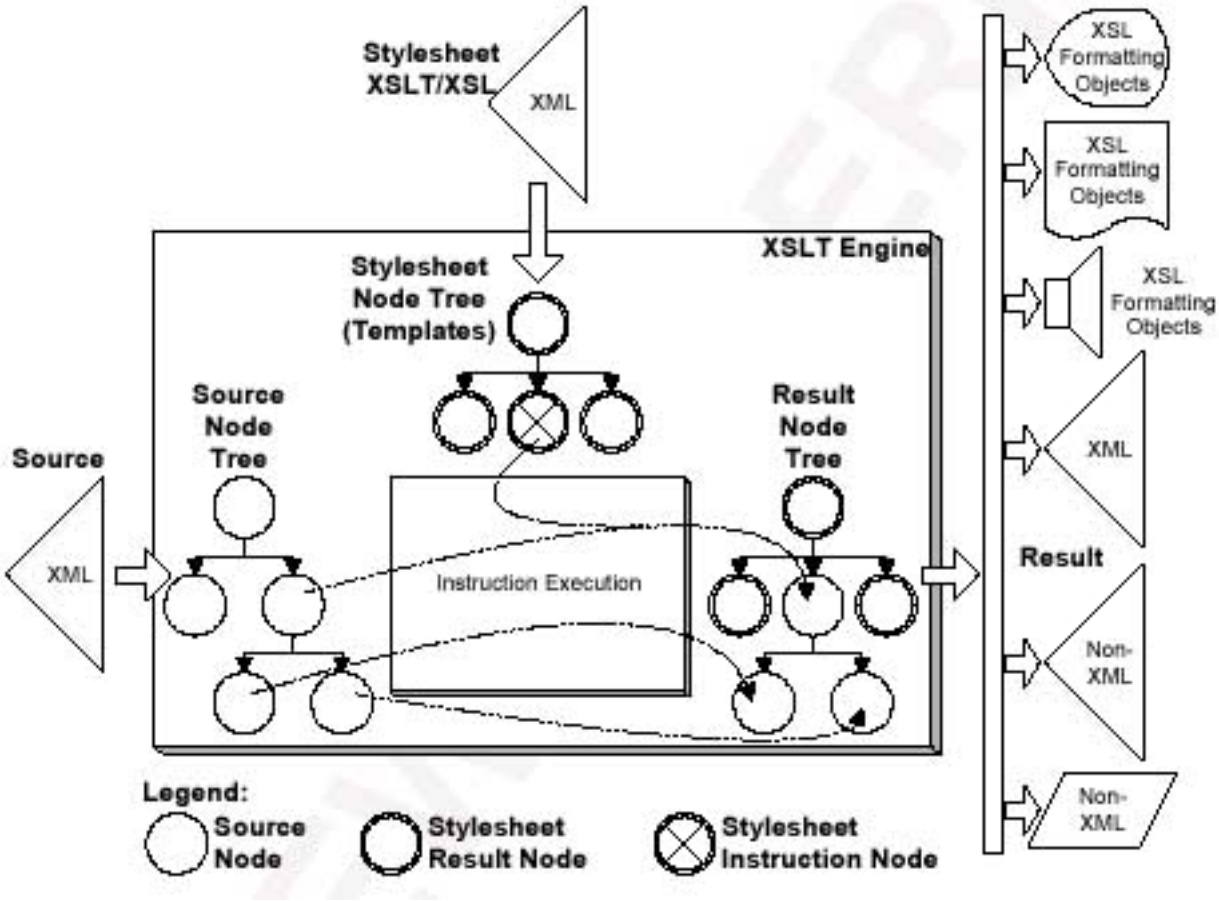
Démo XSL (XML\XSL Visualiseur)

Démo XSL (stock index)



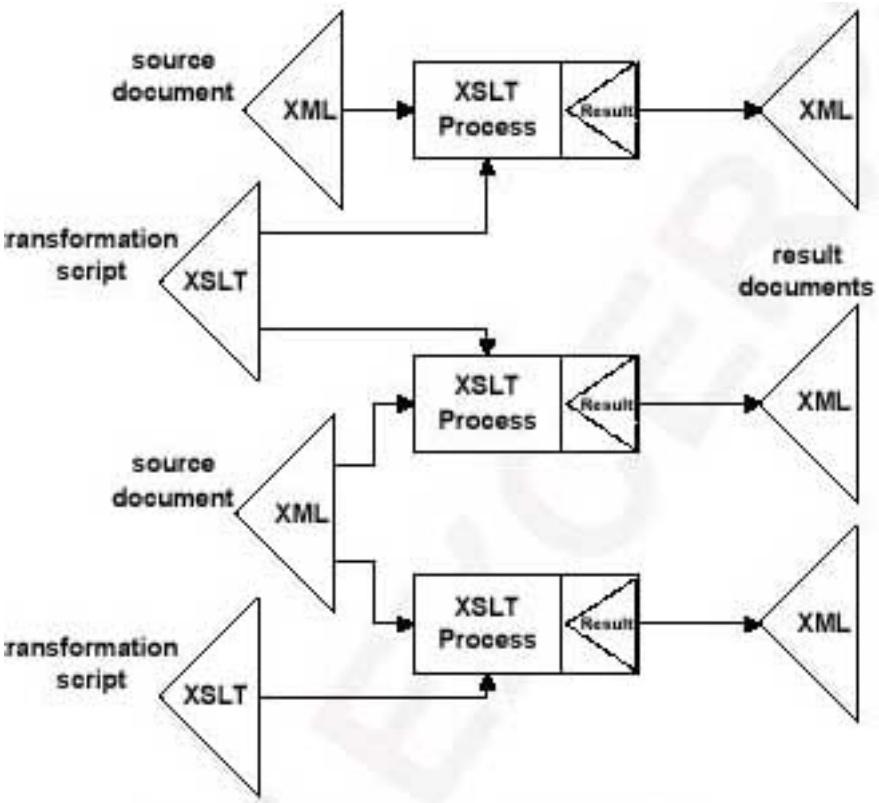
Architecture XSLT

Architecture générale



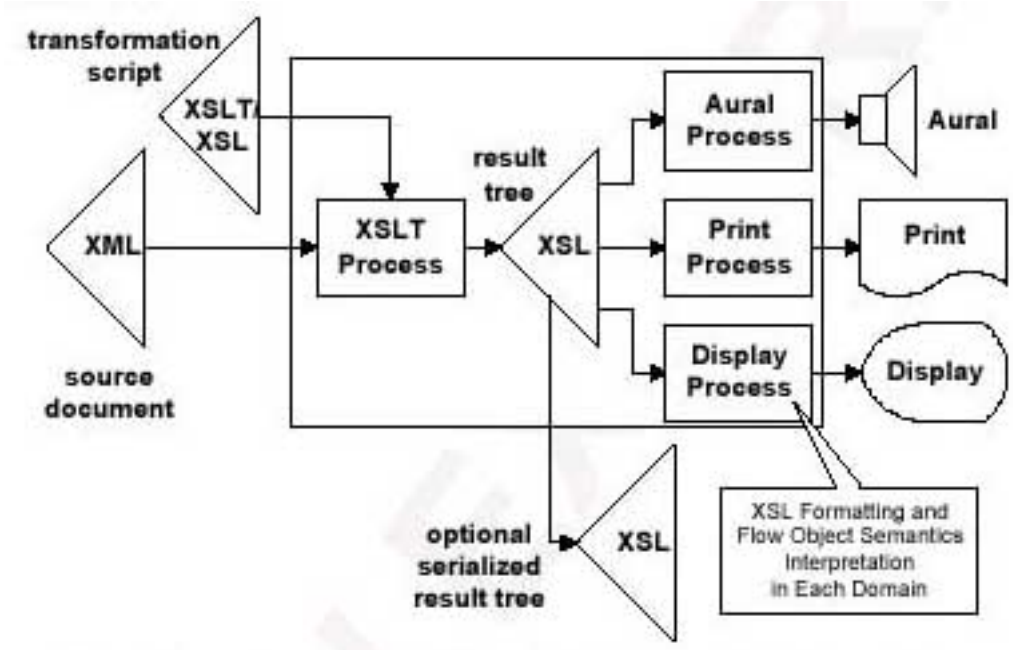
Les différentes transformations possibles:

1-Transformation d'un document XML en un document résultant XML



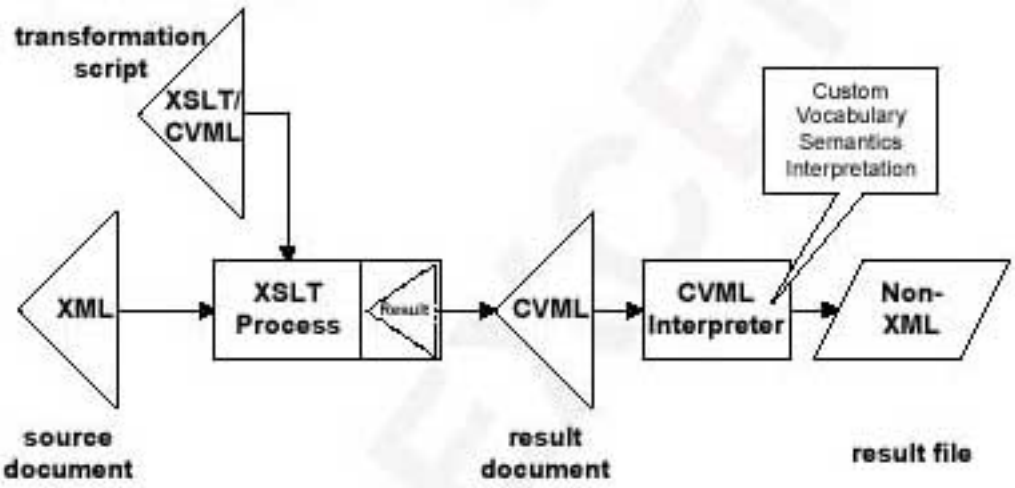
Transformation from XML to XML

2- Transformation d'un document XML en un document résultant XSL-FO



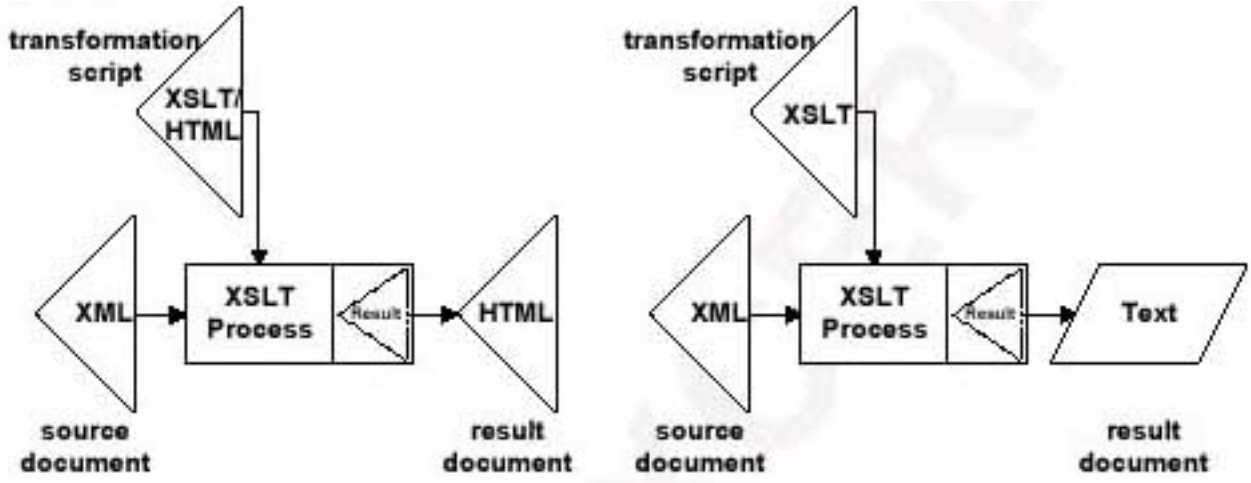
Transformation from XML to XSL Formatting Semantics

3-Transformation d'un document XML en un document résultant non XML



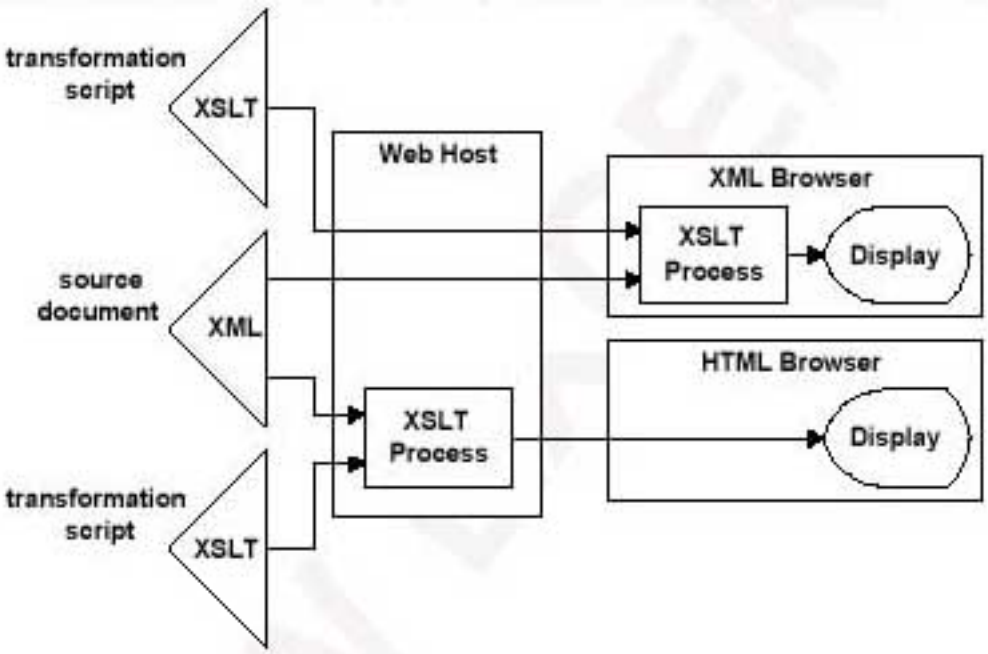
Transformation from XML to Arbitrary Non-XML

4- Transformation d'un document XML en un document résultant non XML



Transformation from XML to Aware Non-XML

Architecture trois Tiers



Server-side Transformation Architecture

Les éléments XSL

éléments	
xsl:stylesheet	L'élément racine d'un stylesheet
xsl:template *match*	Définit une règle de transformation, l'attribut match identifie le noeud ou la règle doit s'appliquer
xsl:apply-templates	Indique au processeur XSL d'appliquer le template
xsl:value-of	Insère la valeur du noeud sélectionné.
xsl:for-each	Applique le template a un ensemble de noeuds.
xsl:comment	Génère un commentaire dans le fichier sortie.
xsl:element	Crée un élément avec le nom spécifié.
xsl:attribute	Crée un nœud attribut et l'attache a son élément dans le fichier sortie.
xsl:copy	Copies le nœud courrant du source dans le fichier sortie.
xsl:if	Permet des template conditionnels.
xsl:pi	Génère une instruction de traitement dans le fichier sortie.
xsl:sort	Trie les éléments
xsl:choose	Procure des test conditionnels multiple avec les éléments xsl:when et xsl:otherwise
etc	

Exemples de Transformations XSL

A. Documents réguliers (répétition d'éléments type, BdD)

A1- Document source XML réguliers transformé en HTML

B. Documents non réguliers (arbre d'éléments en profondeur)

B1- Document source XML non Réguliers transformé en HTML

Utilisation de XSLT

Création de feuilles de style de transformation

Exemples:

- Utilisation de Infoteria XML Style Wizard (XML to HTML)
- Utilisation de Infoteria iXSLT (moteur XSLT)
- Utilisation de Visual XML Transformation

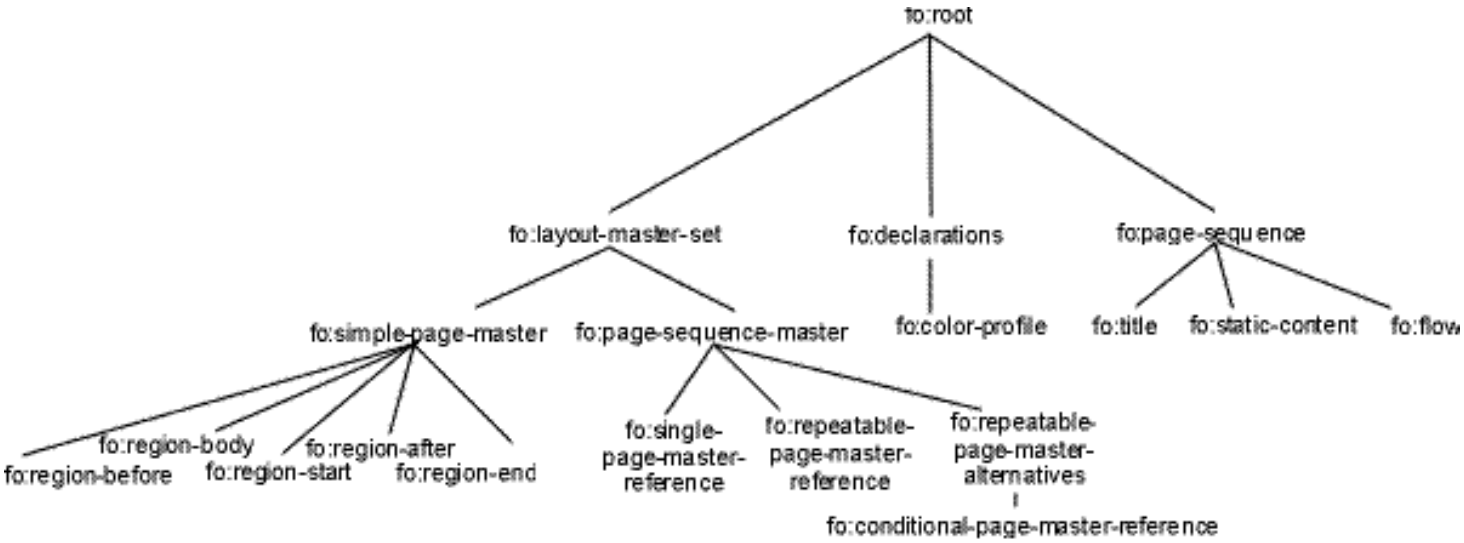
Qu'est ce que XSL-FO?

- Un vocabulaire de formatage permettant la visualisation et l'impression de documents XML
- Un namespace `xmlns:fo="http://www.w3.org/1999/XSL/Format"`

Les éléments XSL-FO

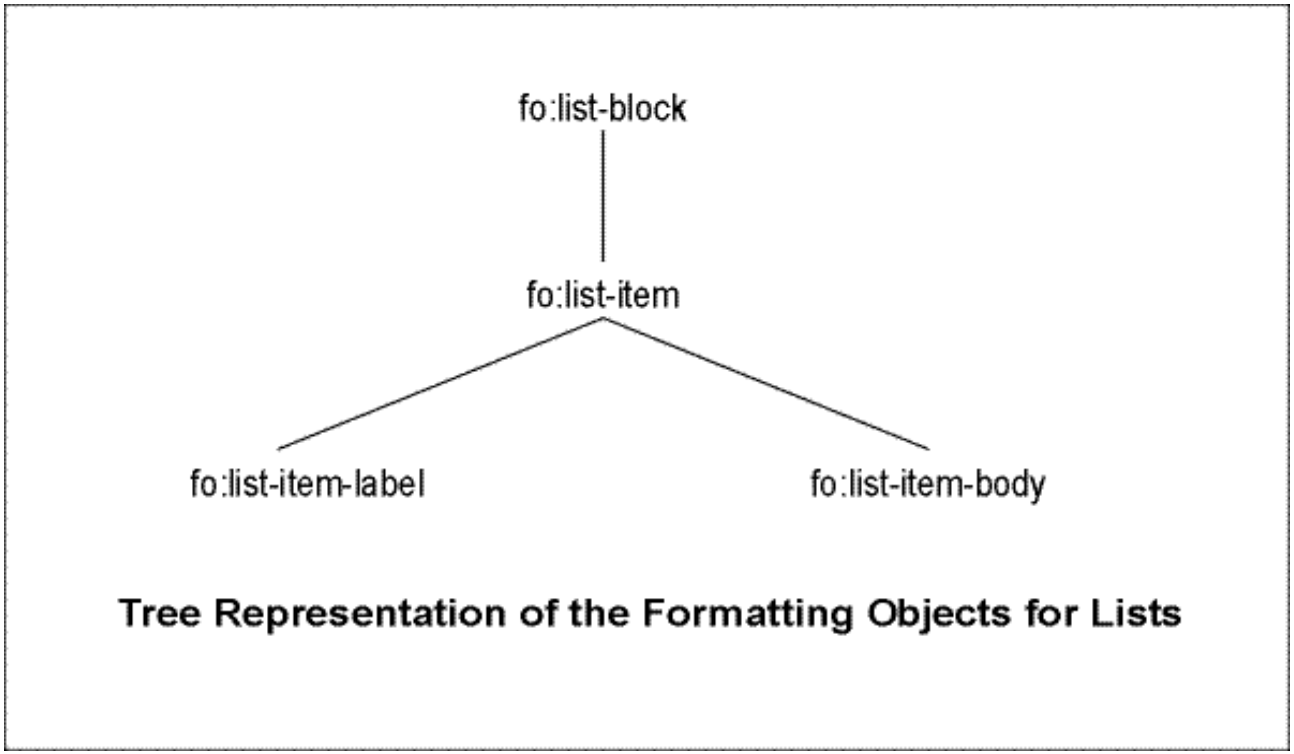
éléments	
fo:basic-page-sequence	Caractéristiques générale d'une page Web (affichage ou impression)
fo:block	Zone contenant des lignes de texte
fo:display-graphic	Zone de niveau block contenant un graphique <img>
fo:display-link	Lien produisant une zone de niveau block <A>
fo:list-block	Zone de niveau block contenant une liste
fo:inline-sequence	Regrouper des objets d'affichage de niveau block
fo:list-item	Élément d'une liste. Permet de redéfinir ses propriétés.
fo:list-item-body	Le corps d'un élément d'une liste <dt> body
fo:list-item-label	Label d'un élément d'une liste <dl>label
etc ...	

Éléments permettant de décrire une page:

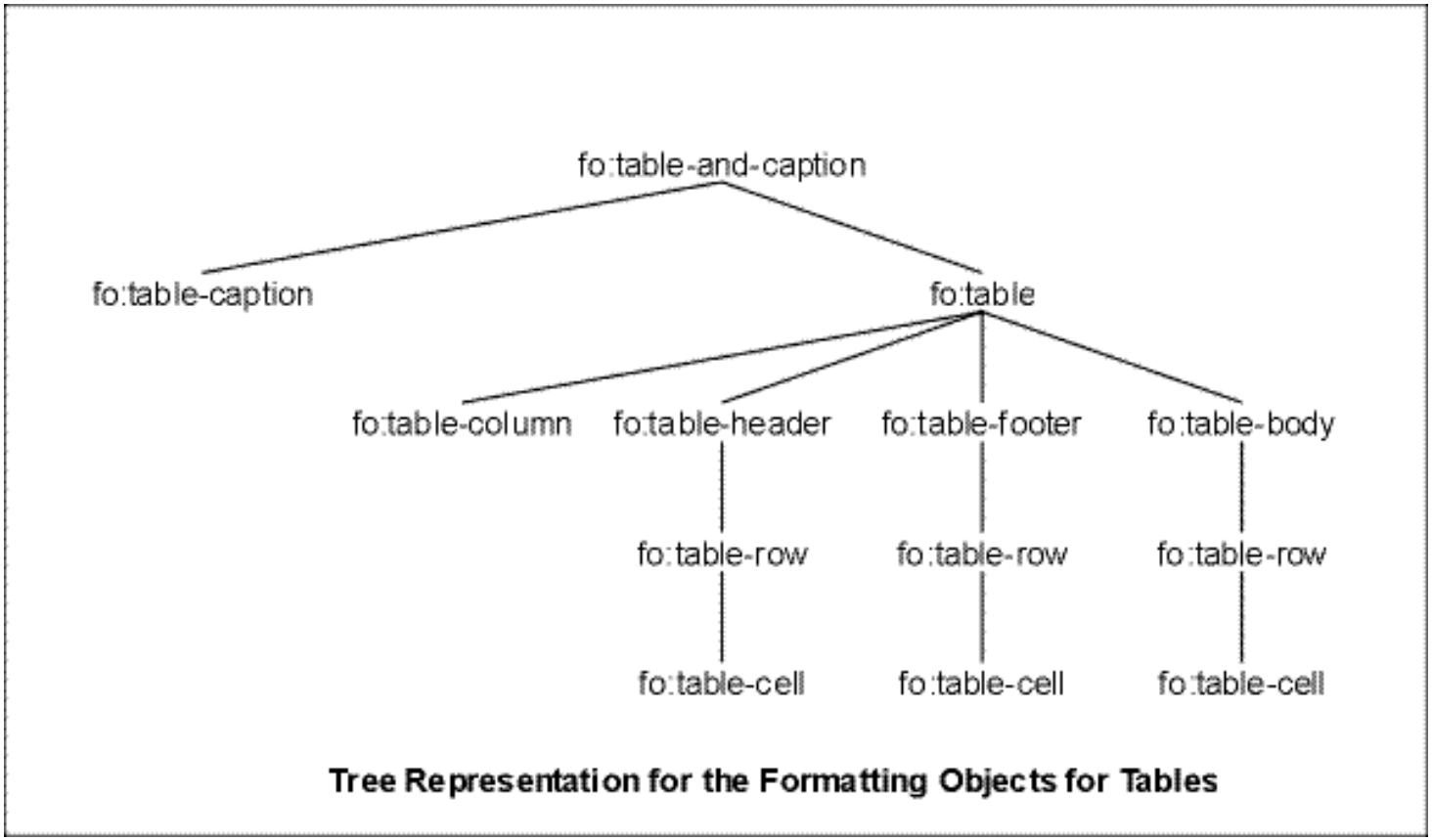


Tree Representation of the Formatting Objects for Pagination

Eléments permettant de décrire une liste

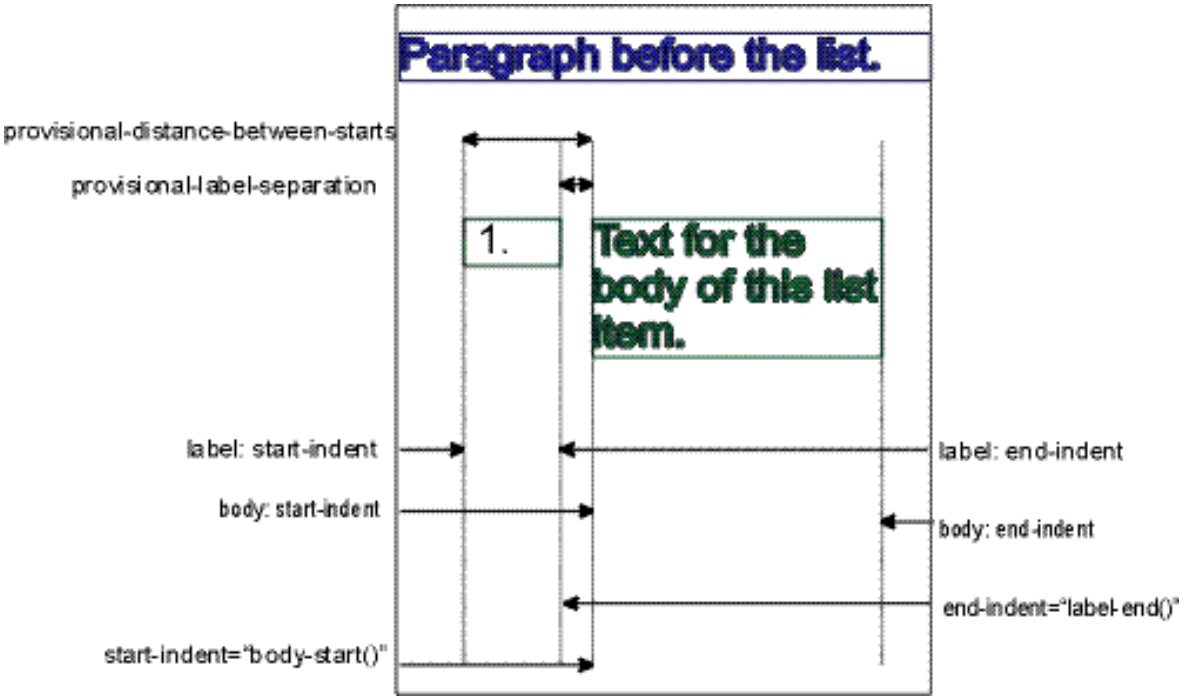


Éléments permettant de décrire une table:



Tree Representation for the Formatting Objects for Tables

Attributs de formattage


Paragraph before the list.
provisional-distance-between-starts
provisional-label-separation
1.
Text for the body of this list item.
label: start-indent
label: end-indent
body: start-indent
body: end-indent
end-indent="label-end()"
start-indent="body-start()"

Les Transformations XSL

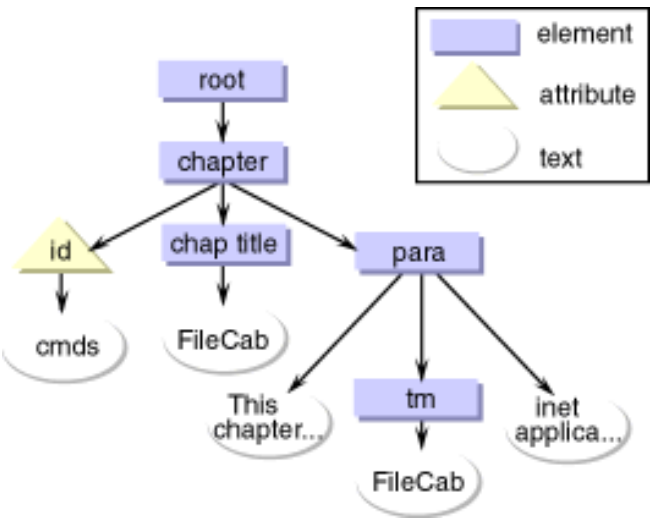
Comment l'utiliser?

- Le document XML indique la feuille de style a appliquer
- La feuille de style est chargée par le parseur XML
- La transformation s'exécute sur le serveur ou sur le client.
- La transformation crée un fichier résultant:
  - HTML: le langage d'affichage des browser actuels
  - XSL-FO: le langage d'affichage des browser compatibles XSL-FO

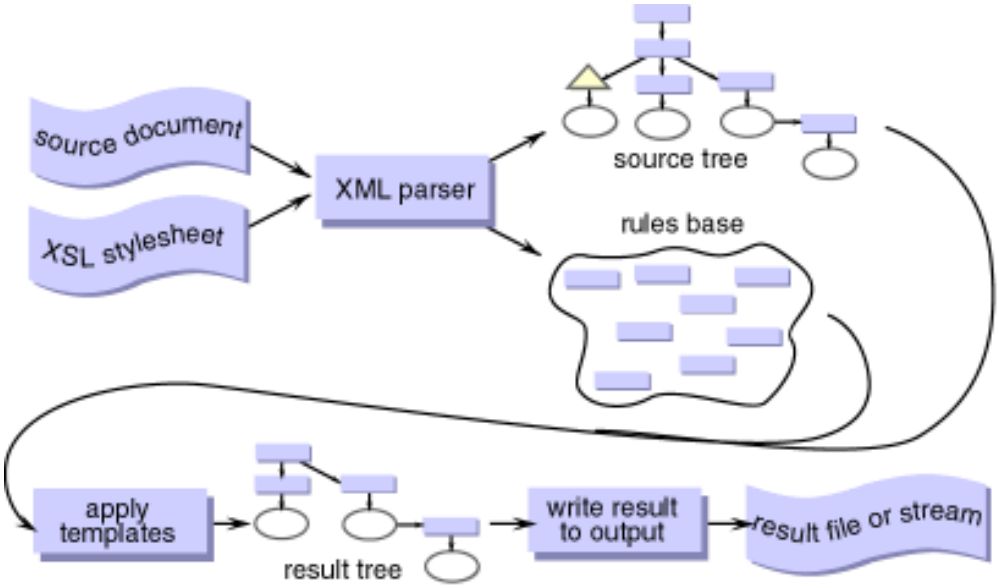
1. Un fichier XML en entrée

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml" href="monstyle.xsl"?>
   <chapter id="cmds">
      <chaptitle>FileCab</chaptitle>
      <para>This chapter describes the <tm>FileCab</tm>inet application.
   </para>
   </chapter>
```

2. Le parseur construit l'arbre DOM



3. Le parseur transforme l'arbre


source document
XSL stylesheet
XML parser
source tree
rules base
apply templates
result tree
write result to output
result file or stream

Exemples de Transformations XSL (XSL-FO)

A. Documents réguliers (répétition d'éléments type, BdD)

A2- Document source XML réguliers transformé en XSL-FO

Comparaison HTML/CSS et XSL-FO:

Objet de mise en forme	HTML/CSS	XSL-FO
Page de document	<HTML>	<fo:page-sequence>
Liste d'éléments	<UL>	<fo:list-block>
Élément	<LI>	<fo:list-item>
Entête de l'élément		<fo:list-item-label>
Corps de l'élément		<fo:list-item-body>

B. Documents non réguliers (arbre d'éléments en profondeur)

B2- Document source XML non Réguliers transformé en XSL-FO

Comparaison HTML/CSS et XSL-FO

Objet de mise en forme	HTML/CSS	XSL-FO
Page de document	<HTML>	<fo:page-sequence>
Bloc de type paragraphe	<DIV>	<fo:block>
Bloc de type caractère	<SPAN>	<fo:inline-sequence>

Travaux Pratiques récapitulatif: Répertoire

XML Schema

- Un Schema XML définit la grammaire d'une classe de document XML
- Plus avantageux qu'une DTD
- Les DTD permettent de définir uniquement le contenu d'un élément (un autre élément ou une chaîne de caractères)

un Schema XML permet:

- d'utiliser une syntaxe XML
- de typer les données (tel élément est un entier, décimal, booléen, string, id, une URL, etc.
- de spécifier les namespaces extensible)
- de valider un document XML avec un parseur (comme avec une DTD)

XML Schema est une Candidate Recommandation du W3C

Déclaration de type d'éléments:

- Les simpleType n'ont ni fils ni attributs.

  `String,boolean,float,double,decimal,binary,uri,id,integer,negative-integer,long,date, TimeDuration,etc.`
- Les complexType ont des éléments fils, du contenu et des attributs

Déclaration d'attributs:

- Toutes les déclaration d'attributs doivent référencer des simple Type.

Les éléments XML Schema :	Les attributs XML Schema
annotation : Déclare un commentaire **attribute :** Déclare un attribut **complexType:** Déclare le type d'un élément ses éléments fils et son contenu documentation **element:** Déclare un élément enumeration : field group **Import:** Importation d'une Schema **include :** Inclusion d'une Schema key : keyRef : **maxExclusive :** **pattern :** déclaration d'un masque restrictions selector **simpleType:** Déclare le type d'un attribut ou le type et conditions d'un élément sans descendants ni attributs. unique	abstract: content : derivedBy : equivClass : block : final: **fixed** **maxOccurs** **minOccurs** name : null : nullable : order : ref : refer : **schemaLocation** : localisation d'un Schema (Namespace/fichier) **base** **type** **value** **version**

Structure d'un Schema XML

Un élément racine

<**schema**> (incluant la déclaration du namespace schema)
```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema"
```

Un arbre d'éléments et leurs attributs:

- **élément** :
  Déclaration de l'élément et son type
  - Type existant :`<xsd:element name="libelle" type="xsd:string"/>`
  - Type nouveau :`<xsd:element name="billTo" type="Adresse"/>`
- **attribut** :
  Déclaration de l'attribut et son type
  - Type existant:`<xsd:attribute name="orderDate" type="xsd:date"/>`
  - Type nouveau :`<xsd:attribute name="partNum" type="Numero"/>`
- **simpleType**:
  Déclare un type d'attribut ou d'élément sans descendants ni attributs.
  - Type simple existants (spécifies dans XML shema):
  `String,boolean,float,double,decimal,binary,uri,id,integer,negative-integer,long,date, TimeDuration,etc.`
  - Type simple nouveaux: définis par dérivation de simple types existants: base="type spécifié dans XML shema"

```
    <xsd:simpleType name="Numero" base="xsd:string">
     <xsd:pattern value="/[A-Z]{3}"/>
     </xsd:simpleType>
```

- **complexType:**
  Déclare le type d'un élément; ses éléments fils et son contenu:

  - **Référencé** par un nom de type name =" nom de type"
  EX: l'élément suivant dont le type est Adresse a trois éléments et un attribut "country":

```
  <xsd:complexType name="Adresse">
    <xsd:element name="rue" type="xsd:string"/>
    <xsd:element name="zip" type="xsd:decimal" />
    <xsd:element name="pays" type="xsd:string"/>
    <xsd:attribute name="country" type="xsd:NMTOKEN" fixed="US"/>
  </xsd:complexType>
```

  - **Anonyme** (évite d'être référencé)
  Ces types n'ont pas de "type=" dans la déclaration de l'élément et sont suivi par un Type anonyme:
  Ex: L'élément item est un complexType anonyme ayant comme élément productName, quantity, price, et shipDate, et un attribut partNum.
  L'élément quantity est un simpleType anonyme dérivé d'un entier dont les valeurs varient de 1 a 99.

```
<xsd:complexType name="Items">
   <xsd:element name="item">
      <xsd:complexType>
        <xsd:element name="productName" type="xsd:string"/>
        <xsd:element name="quantity">
           <xsd:simpleType base="xsd:positive-integer">
           <xsd:maxExclusive value="100"/>
           </xsd:simpleType>
         </xsd:element>
         <xsd:element name="price" type="xsd:decimal"/>
         <xsd:element name="shipDate" type="xsd:date" minOccurs='0'/>
```

```
            <xsd:attribute name="partNum" type="Sku"/>
         </xsd:complexType>
      </xsd:element>
   </xsd:complexType>
```

- **Occurrences des éléments et attributs**
  - minOccurs=0. (valeur par de minOccurs =1)
  - maxOccurs:1 ou *

```
<xsd:element name="item" minOccurs="0" maxOccurs="*">
```

XML Schema: le modèle de contenu

Les schemas permettent de definir le contenu des elements (vide, textuel ou mixte)

- Complex Type vide (élément sans contenu)

```
<price currency='EU' value='423.46' />

<xsd:element name='price'>
   <xsd:complexType content='empty'>
   <xsd:attribute name='currency' type='xsd:string' />
   <xsd:attribute name='value' type='xsd:decimal' />
  </xsd:complexType>
</xsd:element>
```

- Complex Type textOnly (élément avec contenu uniquement)

```
<price currency='EU'>423.46</price>

<xsd:element name='price'>
    <xsd:complexType content='textOnly'>
        <xsd:attribute name='currency' type='xsd:string' />
    </xsd:complexType>
</xsd:element>
```

- Complex Type mixte (élément avec contenu et élément fils)

```
  <salutation>  Cher Monsieur
     <name>Robert Smith</name>
  </salutation>

<xsd:element name='salutation'>
   <xsd:complexType content='mixed'>
      <xsd:element name='name' type='xsd:string'/>
   </xsd:complexType>
</xsd:element>
```

Importer des Schemas dans un Schema

Include

- L'élément `include` et son attribut `schemaLocation` permettent d'inclure un Schema
- Le `targetNamespace` du Schema inclu doit être identique au `targetNamespace` du Schema hôte.(les deux schemas sont dans le même Namespace)

Exemple:

```
<schema targetNamespace="http://www.example.com/IPO"
              xmlns="http://www.w3.org/2000/08/XMLSchema"
              xmlns:ipo="http://www.example.com/IPO">

<!-- include address constructs -->
<include
schemaLocation="http://www.example.com/schemas/address.xsd"/>

<!-- le targetNamespace de address.xsd doit être identique a celui
du schema hote-->
</schema>
```

Exemple d'include

Import

- L'élément `import` permet d'importer un schema apartenant a un autre namespace.
- L'élément `import` et ses attributs `schemaLocation` et `namespace` permettent d'importer le Schema
- le schema importé a un `targetNamespace` différent.
- Cette importation permet la validation d'instance utilisant des multiNamespaces

```
<schema targetNamespace="http://www.example.com/Report"
xmlns="http://www.w3.org/2000/08/XMLSchema"
xmlns:r="http://www.example.com/Report"
xmlns:xipo="http://www.example.com/IPO">

<!-- instance -->

<import namespace="http://www.example.com/IPO"
schemaLocation="http://www.example.com/IPO"/ipo.xsd>
```

Exemple d'import

Associer un XML Schema a une instance XML

Permettre la validation d'une instance XML contre un schema

- Indiquer le nom du fichier du Schema
  l'attribut **noNamespaceSchemaLocation** mentionné sur l'élément racine du document
- Indiquer le nom du Namespace et le nom du fichier du Schema
  l'attribut **schemaLocation** mentionné sur l'élément racine du document, associe la paire
  - Namespace
  - Schema

Exemple sans Namespace:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <dossier
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="dossier.xsd">
  <!-- l'arbre d'élément de l'instance -->
  </dossier>
```

Exemple avec un Namespace:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <root
     xmlns="http://www.example.com/Report"
     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
     xsi:schemaLocation="http://www.example.com/Report
                         http://www.example.com/Report.xsd">
 <!-- l'arbre d'élément de l'instance -->
</root>
```

Exemple Schéma: Fichier de Commande

Démo: Validation d'un Schema (XML Schema Validator W3C)
Validation d'une instance contre son Schema : (XML Schema Validator W3C)
- Schema, instance.

Implémentations:

- XSV : XML Schema Validator en ligne du W3C (alpha); idem self-installing version of XSV for WIN32.
- Free, XML Schema validator d' Oracle: C, C++ and Java versions.
- **20000407, version of XML Schema**

- A Conversion tool from DTD to XML Schema in perl; open source.
- XML Schema validator du projet Apache
- XML Schema (and others) aware editor/validator de XML Spy

Travaux Pratiques

Exercice de synthèse

- Créer un document XML bien formé de type Répertoire:
  - composé de fiches (identifiant) incluant chacune:
  - nom, prénom, téléphone (localisation), email
- Créer une DTD pour valider le document
- Créer un Schema pour valider le document
- Créer un feuille de style CSS permettant de l'afficher ce XML
- Créer un feuille de style permettant de l'afficher en HTML (avec XSLT)
- Créer un feuille de style permettant de l'afficher en XSL-FO (avec XSLT)
- Transformation du fichier XML répertoire (les éléments en attributs)

Solution exercice de synthèse :répertoire
Autre exercice de synthèse : sales (HTML et SVG)

W3C

previ conte chang next

XForms 1.0: Data Model

La prochaine génération des formulaires sur le Web.

XForms permet:

- Meilleure intéraction entre les lecteurs/visiteurs/acheteurs et les auteurs/serveur/vendeurs et le programmes (CGI, etc)
- des formulaires Web plus riches permettant plus de flexibilité et de richesse d'interaction
- XForms est le successeur des formulaires HTML, et peut être utilisé dans tous les documents XML.

XForms est un Working Draft

- XForms: une DTD http://www.w3.org/TR/xhtml-forms1/DTD/xhtml-xforms1.dtd
- XForms: Un Namespace http://www.w3.org/2000/xforms

XForms : séparer les datas et la présentation

- Un *data model* définissant les contraintes entre et sur les data.
- L' *interface utilisateur*: des contrôles de présentation lies au data model (a venir)

diagram showing user
interface, data model, instance data and XML encoding

Les types de données

String:
Ex d'une string de 5 caractères: `<string name="postalcode"> <mask>ddddd</mask> </string>`
Nombres:
Ex d'un entier non nul: `<number name="count" min="1" integer="true"/>`
Ex d'une valeur non-négative en British Pounds: `<money name="price" currency="GBP" decimals="2" min="0"/>`

D**ates**:
Exemple d'une date de naissance: `<date name="birth" max="now"/>`
Exemple d'une date d'expiration de carte de crédit:
`<date name="expires" precision="months" min="now" max="+P4Y"/>`

Calculs
le formulaire inclure des valeurs qui sont calculées depuis d'autres valeurs de champs:
Ex: `<currency name="totalPrice" calc="sum(lineItem, quantity * price)"/>`
Cet exemple somme le produit des valeurs `quantity` et `price` avec `lineItem`.

Typage des données (more)

Exemple de Xforms

XML Signature

Signer électroniquement des ressources XML du Web

Signature Syntax and Processing

- Candidate Recommendation (W3C / IETF) -31 octobre 2000.
- Associer le contenu d'une URL a une clé cryptographique
- Authentification, intégrité et la non répudiation

XHTML1.0

Extensible HyperText Markup Language

Une re-formulation de HTML 4.0 en une application XML 1.0
XHTML1.0 est une Recommandation du W3C

Un document conforme a XHTML1.0 doit:

- Valider avec une de ses trois DTD (loose, strict,frameset)
- Avoir <html> comme élément racine
- designer le Namesapce XHTML dans l'élément racine xmlns="http://www.w3.org/1999/xhtml"
- avoir une Déclaration de DOCTYPE avant l'élément racine

La sémantique des éléments et de leurs attributs est celle définit dans la Recommandation du W3C pour HTML 4

XHTML 1.0 a déprécie l'attribut "name" des éléments : a, applet, form, frame, iframe, img, et map

Bénéfices:

- Documents bien formes et valides
- Intégration possible d'autres langages (SVG.MathML, etc.)
- Introduction de nouveaux éléments

Exemple simple d'un document XHTML 1.0

Exemple d'intégration de MathML dans un document XHTML 1.0

Exemple d'intégration de XHTML 1.0 dans un autre XML namespace

---

XHTML1.1

Un document conforme a HTML 1.1 doit:

- Valider avec sa DTD
- Avoir <html> comme element racine
- designer le Namesapce XHTML dans l'élément racine
  `xmlns="http://www.w3.org/1999/xhtml"`
- avoir la Déclaration de DOCTYPE avant l'élément racine
- XHTML1.1 est une Recommendation du W3C

XHTML 1.1 a déprécié tous les attributs de style
XHTML1.1 est modulaire, il est basé sur les modules suivants

Structure, Basic Text, Hypertext, List, Applet, Presentation, Edit, BDO, Forms, Tables, Image, Image Map, Intrinsic Events, Metainformation, Scripting, Stylesheet, et Link comme défini dans XHTMLMOD une Candidate Recommendation du W3C.

Puis prochainement XHTML2.0 ....

Example d'un document XHTML 1.1

Validateur XHTML1.0

W3C

previ conte chang next

XHTML basic

XHTML Basic document type est un sous ensemble de XHTML 1.1

Il inclut un ensemble de modules minimal pour créer du contenu simple
Clients qui ne supportent pas les fonctionnalité de XHTML (téléphone, phones, PDAs, pagers, et settop boxes).

Un document conforme a HTML basic doit:

- Valider avec sa DTD
- Avoir <html> comme élément racine
- designer le Namesapce XHTML dans l'élément racine
  `xmlns="http://www.w3.org/1999/xhtml"`
- avoir la Déclaration de DOCTYPE avant l'élément racine
  `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN" "xhtml-basic10.dtd">`
- XHTMLbasic est une Recommendation du W3C

XHTML Basic est base sur les modules XHTML suivants:

Structure Module	body, head, html, title
Text Module	abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var
Hypertext Module	a
List Module	dl, dt, dd, ol, ul, li
Basic Forms Module	form, input, label, select, option, textarea
Basic Tables Module	caption, table, td, th, tr
Image Module	img
Meta Information Module	meta
Link Module	link
Base Module	base

Tous ces modules sont définis dans XHTMLMOD

Demo Amaya

Thierry MICHEL

SMIL

SMIL : Synchronized Multimedia Integration Language

- une application XML, et une grammaire DTD
- Produire des présentations multimédia interactives
- Mixer et synchroniser des objets média (textes, images, vidéo, audio)
- SMIL 1.0 W3C Recommandation, 15 juin 1998
  SMIL 2.0 est une Recommandation, 07 Aout 2001 du W3C

Fonctionnalités:

- Décrire le placement géométrique des composants visuels
- Décrire le placement temporel de ces composants
- Associer des hyper liens avec des objets média
- Effets d'activation (évènements temporels, évènements utilisateur)
- Offrir des alternatives de présentation (langage, taille écran, bande passante, soutirages, légendes, etc.)

Bénéfices:

- Lisible par des humains et des machines
  Facile a modifier
  Plate-forme de production indépendant
  Déclaratif (pas de scripts)
- Indépendant des formats de média a intégrer
- les objets média peuvent être styles (CSS, XSL)
- Accessible (légendes, doublage, etc.)

SMIL: Placement et Synchronisation

Les éléments médias SMIL sont de deux types:

- **Médias continus**(durée intrinsèque):
  <Audio>, <video>, <animation>, <textstream>, <ref>
  ```
  <video src="joe-video" region=video/>
  ```
- **Médias discrets**(sans durée intrinsèque):
  <text>, img>
  ```
  <img src="copyrights-joe-video" region=image/>
  ```

attributs éléments media:

```
- synchronisation:begin, end, dur, clip-begin, clip-end, fill(remove,freeze)
- positionnement: region,
- identification: src, id, copyrights, author, alt, title, type, etc.
- caractéristiques (test booléens):System-bitrate, system-caption, ssystems language,
system-screen-size, system-screen-deph, system overdub-or-caption
```

Placement des médias:

- l'élément vide <root layout> indique les dimensions et la couleur du fond de la fenêtre de présentation
  ```
  <root-layout  height="1300" width="600"/>
  ```
- l'élément <region> permet de subdiviser cette fenêtre en autant de régions nécessaires au positionnement des médias.
  ```
  <region id="video" top="5" left"15" height="100" width="300"/>
  ```
- l'attribut région permet de placer le composant dans cette région
  ```
  <video src="joe-video" region=video/>
  ```

Synchronisation des médias:

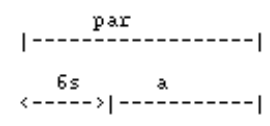
- les éléments container <par> et <seq> permettent de grouper et synchroniser les médias.

  attributs éléments container:

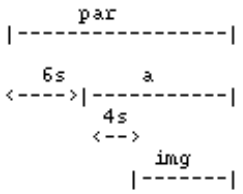
  ```
  - synchronisation:begin, end, dur, endsynch(last, first, IDREF), repeat
  - positionnement: region,
  - identification: id, copyrights, author, title.
  - caractéristiques (test booléens):System-bitrate, system-caption, ssystems language,
  system-screen-size, system-screen-deph, system overdub-or-caption
  ```

1. Élément Parallèle <par>:

```
<par>
  <audio id="a" begin="6s" src="audio" />
</par>
```

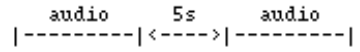



```
<par dur="20s">
<audio id="a" begin="6s" src="audio" />
<img begin="id(a)(4s)" src="image.jpg" fill="freeze"/>
</par>
```



2. Élément Séquence <seq>:

```
<seq>
  <audio src="audio1" />
  <audio begin="5s" src="audio2" />
</seq>
```



Exemple: Structure de base d'un document SMIL
Graphique: Synchronisation des médias

SMIL: Les liens et le contrôle de contenu

Une présentation SMIL n'est pas une simple vidéo, l'utilisateur a le choix d'un itinéraire dans la présentation, d'une navigation hypermédia.

L'élément <A> offre les mêmes fonctions que celui de HTML.
- attributs : `Id, href, show (replace, new, pause)`

L'élément <anchor> offre des fonctions étendues par rapport a celui de HTML.
- attributs : `Id, href, begin et end (spécifient une fenêtre temporelle pendant la laquelle le lien est traversable)`

L'élément <anchor> permet :

- d'associer des liens différents a des rectangles définis dans un composant
- d'associer des liens différents a des intervalles de temps différents d'un composant
- de définir une zone rectangulaire dans la présentation d'un composant comme étant la cible d'un lien en lui associant in id
- de définir une fenêtre temporelle de la présentation d'un composant comme étant la cible d'un lien en lui associant in id

Exemple: différents types de liens

SMIL permet des alternatives de présentation d'un composant en fonction des préférences de l'utilisateur.
en fonction de la taille de l'écran, du langage, de nombre de couleurs, bande passante, etc.

SMIL définit la sémantique des attributs suivants:
`System-bitrate, system-caption, ssystems language, system-screen-size, system-screen-deph, system overdub-or-caption`

Exemple: contrôle de contenu (switch)

Démo SMIL

SMIL version 2.0

- Étendre les fonctionnalités de SMIL 1.0.
- Définir un ensemble de modules réutilisables et définissant les fonctionnalités SMIL.
- Associer un Document Object Model (SMIL DOM).
- Permettre la réutilisation de ces modules (SMIL syntaxe et sémantique) dans d'autres langages XML qui ont besoin de timing et de synchronisation.
  - Téléphones Mobiles WAP/WML langage, ou XHTML-Basic.
  - SMIL Animation est déjà utilise dans Scalable Vector Graphics language (SVG).
  - XHTML+SMIL: intégration de SMIL avec XHTML.

Modularisation en Module

- Animation Module
- Transitions Module
- Linking Module
- Timing Module
- etc.

SMIL2 est un Working Draft du W3C

Exemple de Transition

Exemple d'Animation

Démos: Animation
- rectangle (taille et opacité)

- Démos HTML + SMIL: AC, IE5.5, spice

Graphiques: SVG (Scalable Vector Graphics)

Langage pour décrire des graphiques 2D en XML

Fonctionnalités:

- Inclure des objets graphiques: graphiques vectoriels, formes, images et du texte.
- Ces objets graphiques peuvent être groupés, stylés, transformés et animés

Avantages:

- Taille des fichiers réduite
- Format textuel: permet de faire des recherches et les textes sont accessibles
- Vectoriel: aucune dégradation (imprimer/retailler/zoom)
- Pas de multiple versions d'un même graphique: haut niveau de détails a des niveaux de zoom différents.
- Les objets peuvent être styles (CSS, XSL)
- Extensible: supporte XML-Namespaces (pour insérer d'autres type de données XML)
- SVG 1.0 est une Recommendation du W3C.

Exemple SVG: deux groupes comprenant deux rectangles et de couleur différentes

Exemple SVG: deux groupes comprenant deux ellipses couleur différentes

Exemple SVG: deux polygones

Exemple:Insertion de SVG dans un document XML

Démos SVG:
Carte, W3C, tigre, trèfle
Picasso, poisson, lion, papillon, filtre marbre, textes

Démos: Animation SVG
- rectangle (taille et opacité)
- Carte animée
- Ovales animes
- W3C lumières spots
- Turbulences
- Horloge

Demo Amaya -SVG

Mathématiques: MathML

Langage pour décrire des expressions mathématiques en XML

Fonctionnalités:

- Encode la présentation de notations mathématique et leur contenu sémantique.

Avantages:

- Facilite l'usage et le re-utilisation de contenu mathématiques sur le Web
- Compatible avec des applications utilisant la sémantique (logiciels scientifiques ou synthétiseurs vocaux).
- Format textuel pour la communication de machine a machine
- Peut être style (CSS, XSL)
- Accessible
- MathML 1.0 est une Recommandation du W3C
- MathML 2.0 est une Recommandation du W3C

Exemple Représentation par des balises de présentation

Exemple: Représentation par des balises de sémantique

Demo MathML -Amaya

RDF (Resource Description Framework)

Fonctionnalités:

- Langage pour décrire des meta-datas en XML
- Utiliser les meta-data "données sur les données" pour décrire des données sur le Web
- RDF- Model and Syntax Specification est une Recommandation du W3C, 22 fev 1999.

Avantages:

- Cataloguer pour décrire le contenu et les relations de documents
- Décrire les propriétés intellectuelles et copyright de pages Web
- Exprimer la "privacy preferences" des utilisateurs et celle des sites Web
- Décrire des collections de pages qui représentent un seul document logique
- Rechercher des documents sur le web (Moteurs de recherche)

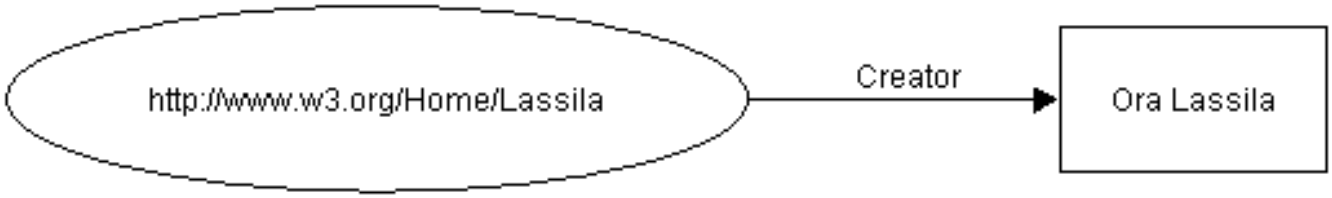
RDF (Modèle de Base)

Le modèle RDF permet d'associer des attributs aux ressources du Web

Trois types de données (Objets) :

- **Les ressources: (**Resources) page Web, morceau de page, fragment, site web etc . identifié par son URL
- **Les propriétés:** (Property)caractéristiques ou attributs pour décrire la ressource associe a sa valeur. La façon d'exprimer les caractéristiques des ressources est définie dans "RDF Schema" .
- **Les déclarations**:(Statement): Le triplet compose d'une ressource, d'une propriété et d'une valeur de cette propriété.

Exemple de modélisation d'une déclaration DRF *Ora Lassila est le créateur de la ressource "http://www.w3.org/Home/Lassila".*



Le document complet XML document décrivant le shema les propriétés ci dessus:

Avec un Namespace par défaut

```
<?xml version="1.0"?>
<RDF
  xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:d="http://description.org/schema/">
  <Description about="http://www.w3.org/Home/Lassila">
    <d:Creator>Ora Lassila</d:Creator>
  </Description>
</RDF>
```

Syntaxe:

<RDF> élément permettant a un processeur de repérer le début d'un bloc de déclarations RDF
<Description> permet d'identifier la ressource. une liste d'éléments permet permet de d'en définir les propriétés.

Les domaines de noms permettent d'identifier les propriétés appartenant a tel ou tel autre Schema. La nature est la sémantique des propriétés sont définies dans le RDF Schema.

```
<?xml version="1.0"?>
<RDF
  xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://description.org/schema/"
  xmlns:secu="http://securitesociale.org/schema/">
  <Description about="http://www.w3.org/Home/Lassila">
    <dc:Creator>Ora Lassila</dc:Creator>
    <secu:Numero>1750422541529</secu:Numero>
  </Description>
</RDF>
```

W3C

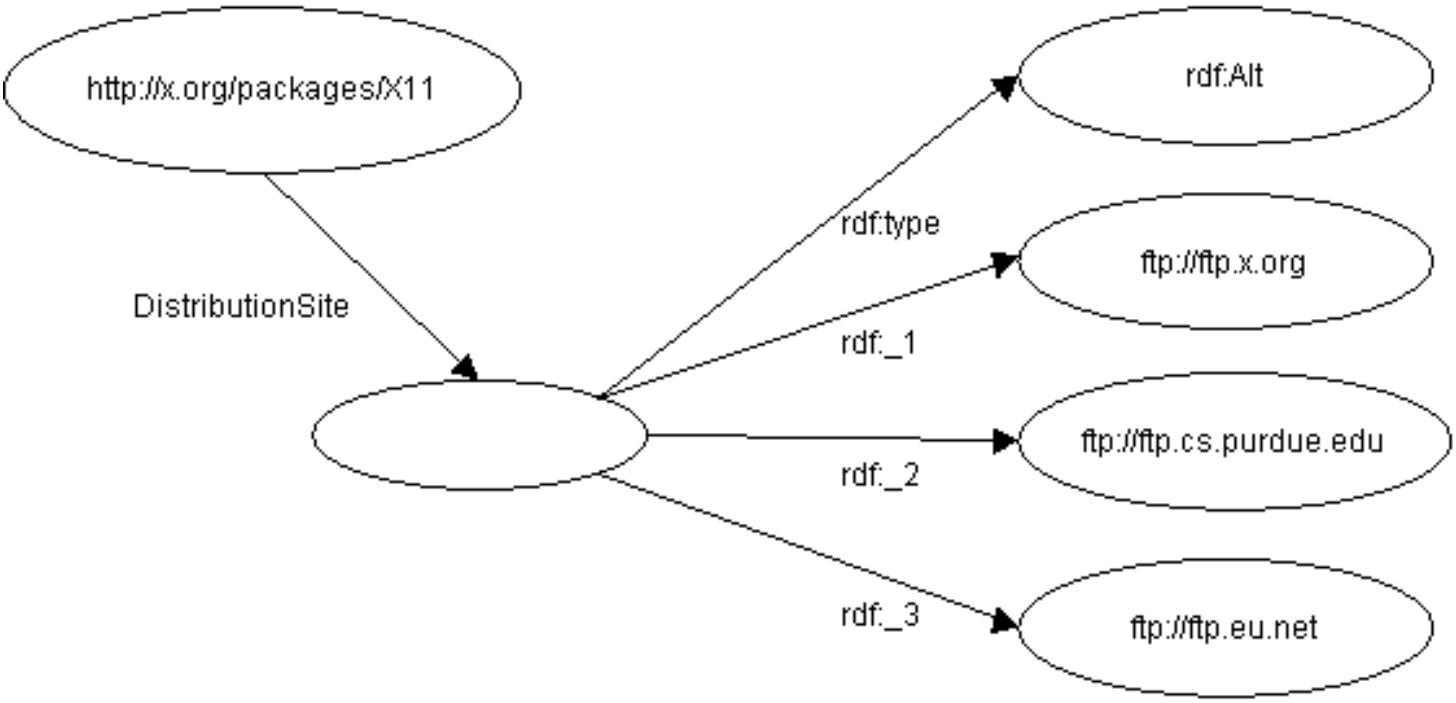
previc conte chang next

RDF: Les Collections

Les collections (container) permettent de manipuler des collections de ressources lorsqu'une propriété s'applique a plusieurs Objets.

- Bag : liste non ordonnée de ressources. Une propriété a plusieurs valeurs mais l'ordre est indifférent
- Seq :liste ordonnée de ressources. Une propriété a plusieurs valeurs mais l'ordre est important
- Alt :liste de ressources représentant des alternatives. La propriété peut prendre une des valeurs

Exemple liste de téléchargement (Alternative)
Le code source de X11 peut être télécharge a ftp.x.org, ftp.cs.purdue.edu, or ftp.eu.net.



s'écrit RDF/XML:

```
<rdf:RDF>
  <rdf:Description about="http://x.org/packages/X11">
    <s:DistributionSite>
     <rdf:Alt>
        <rdf:li resource="ftp://ftp.x.org"/>
```

```
        <rdf:li resource="ftp://ftp.cs.purdue.edu"/>
        <rdf:li resource="ftp://ftp.eu.net"/>
        </rdf:Alt>
        </s:DistributionSite>
   </rdf:Description>
</rdf:RDF>
```

Exemples:

- **Exemple RDF composé**
- **Exemple: Dublin Core Metadata:**

  Propriétés descriptives de données documentaires (Title, Author, Subject, Description, Publisher, Date, Format, Language, etc.)
  Le Schema Dublin Core
  (http://purl.org/DC/documents/rec-dces-19990702.htm)

- **Exemple: metadata et SMIL**
- **Exemple: RDF et Micropayment**

Demos

- Démo (online) RDF et les photographies
  (http://jigsaw.w3.org/Yves/Australia/ avec [;text/rdf] a la fin de L'URL)
- Démo: outils de visualisation de Graph RDF.

Conclusion

Le Web aujourd'hui:

HTML: Une technologie simple

- Fournir de l' information a consommer par de humains "machine-readable", "human-understandable".
  (ex: difficultés pour les moteurs de recherche de cataloguer l'information

Le Web de demain:

Base sur des technologies XML, (XHTML, SVG, SMIL, XSL, RDF, MathML, CML, etc.)

- Fournir de l'information a consommer par les humains et les machines: données "machine-understandable".
- Plus de standards
- Plus de traitements automatises et fiable sur le Web (XML/EDI)
- Intégration de types de documents différent
- Généraliser l'accès (computers, laptops, téléphones, TVs, palmpilot, etc.)
- Plus d'accessibilité
- Des moteurs de recherche plus puissant grâce aux metadata

Une architecture pour des applications distribuées.

Appendices

Logiciels

Nous citons ici un certain nombre de logiciels, tous disponibles gratuitement pour évaluation. Pour certains d'entre eux, une version commerciale est proposée avec une offre de services associés. Cette liste n'est ***absolument pas exhaustive***, et notre choix est largement arbitraire. Au moins les logiciels qui y figurent sont-ils maintenus et documentés.

Parseurs

Ælfred

Implémentation en Java, très compacte, d'un parseur " dtd-aware " (traite la DTD lorsqu'elle existe). Le parseur est non validant : la DTD est notamment utilisée pour déterminer la présence et la valeur des attributs implicites, mais il n'y a pas de contrôle de validité au sens strict. Résolution des entités externes par URL. Une API SAX est intégrée au parseur. Du fait de sa compacité, ce parseur est recommandé par ses auteurs pour l'implémentation d'applets. Version gratuite pour expérimentations et version commerciale.

Origine :

Microstar Software Ltd. (Canada)

DataChannel XML Parser (DXP)

Parseur *validant* écrit en Java. DXP est proposé avec trois interfaces : ligne de commande, et APIs SAX et DOM. Supporte un catalogue de FPI pour la résolution des entités externes. Recommandé pour l'implémentation des applications " coté serveur ". Version gratuite pour expérimentation (45 jours) et version commerciale proposée avec d'autres outils XML.

Origine :

DataChannel (USA)

Expat

Paseur *non validant*, écrit en C, de haute performance. Disponible gratuitement sous licence publique Mozilla. Source C disponible.

Origine :

James Clark

Lark et Larval

Lark est un parseur *non validant* écrit en Java. Son api, non conforme au DOM, permet néanmoins d'accéder à l'arbre du document. Il traite toutes les entités internes ou externes, mais les fpi contenus dans les déclarations public sont simplement passés au handler de l'application. Larval est un package Java, basé sur Lark, qui permet la validation du document selon sa DTD. Source disponible.

Origine :

Tim Bray (Textuality)

Microsoft XML Parser for Java (MSXML)

Parseur *validant* écrit en Java. Son api est proche de DOM, l'intention déclarée de Microsoft étant de se conformer au standard dès que celui-ci sera officiellement publié. Le parser est intégré dans la distribution de Internet Explorer depuis la version 4 de ce dernier.

Origine :

Microsoft

IBM XML for Java

Parseur *validant* écrit en Java. Disponible avec des api SAX et DOM. En fait, XML for Java est plus qu'un parseur puisqu'il propose également des classes permettant de *générer* un document XML conforme à une DTD donnée. Il est distribué avec un certain nombre d'applications de démonstration, notamment un processeur de pointeurs XML. License d'évaluation gratuite. License commerciale proposée.

Origine :

IBM

XP

Parser *non validant*, écrit en Java. Traite toutes les entités internes ou externes. Source disponible.

Origine :

James Clark

APIs

SAX 1.0.

API définie par un groupe de développeurs réunis par la liste de discussion xml-dev, coordonnés par

David Megginson. Bon nombre des parseurs mentionnés ci-dessus disposent d'un driver SAX (éventuellement développé par un tiers), permettant ainsi au programmeur d'application d'utiliser toujours la même API. L'API elle-même est implémentée en Java. Elle est disponible gratuitement pour toute utilisation, commerciale ou non.

Distribution :

David Megginson

FREE-DOM

Implémentation gratuite, en Java, de l'api DOM, par Don Park. Free-dom est basé sur SAX. Un premier driver permet de faire le lien entre Free-dom et SAX et un second entre SAX et un parseur particulier. Toutefois, le développement de drivers "directs" pour certains parseurs, notamment Ælfred et MSXML, est prévu.

Origine :

Don Park

Environnements de développement, boites à outils

XML en Python

Peut-être pas le plus rapide à l'exécution (quoi que...) mais surement le plus convivial pour le programmeur et le plus rapide en temps de développement. Offre une API Python au parseur expat de James Clark, et une librairie de manipulation XML. Il faut souligner qu'un langage de haut niveau comme Python, doté d'une très riche librairie de manipulation de chaines de caractères, de dictionnaires, etc. se prête particulièrement bien à l'écriture d'applications XML.

Origine :

Groupe de travail XML / Python

Java Project X: Java Services for XML Technology

L'environnement complet de manipulation XML en Java proposé par Sun. Disponible gratuitement à condition de s'enregistrer comme "Java Developer". Le parseur propose les deux APIs DOM et SAX. Support des caractères Unicode, codages UTF-8 et UTF-16, en sus des codages classiques ISO-8859-X.

Origine :

Sun Microsystems

SAXON

Librairie (gratuite) de classes Java supportant un certain nombre de traitements xml. Saxon est particulièrement adapté à la programmation d'applications réalisant des transformations xml / xml ou xml / html.

Auteur :

Michael Kay

DAE SDK

Boite à outils générale SGML, XML et DSSSL écrite en Java. Inclut un parseur validant XML, un constructeur de bosquets, le traitement des requêtes SDQL, et une machine DSSSL pour le formatage des documents SGML ou XML. La machine DSSSL accepte les scripts écrits en Scheme. Produit commercial. Evaluation gratuite sur demande.

Origine :

Copernican Solutions

LT XML

Librairie écrite en C (pour changer un peu !) qui implémente un parseur XML complet et un ensemble de fonctionnalités supplémentaires. Le parseur offre au choix un accès dirigé par les événements ou un accès à l'arbre complet du document, à travers une api C. Les fonctionnalités supplémentaires sont l'extraction de texte du document XML, la recherche d'expressions régulières, les transformations XML / XML, la tokenisation (au sens de l'ingénierie linguistique), le tri d'éléments selon leur contenu, et la transclusion déclenchée par la traversée de liens XLink (tous les liens XLink ne sont pas supportés dans la version 1.0).
LT XML est disponible gratuitement pour les utilisations non commerciales. Source disponible.

Origine :

Henry S. Thompson (HCRC Language Technology Group, University of Edinburgh)

Bases de données xml

PSE/PSE Pro for Java

Système de gestion de bases de données objet, entièrement écrit en Java, interfacé avec le parseur " XML Parser for Java " de Microsoft, permettant la persistence des arbres XML résultant du parsing. C'est un SGBD " personnel ", n'acceptant pas des transactions multiples parallèles. La version " professionnelle " (commerciale) PSE Pro supporte au dire du constructeur des bases de l'ordre de quelques centaines de MO, et permet l'indexation d'objets collection. La version gratuite pour usage personnel PSE n'offre pas de mécanisme d'indexation, et ne supporte que les bases de quelques dizaines de MO.
L'API Java permet notamment de créer une base, d'ouvrir et de fermer une session, et de faire des transactions en lecture et écriture sur les objets. L'accès depuis un script cgi est simple à réaliser et permet de servir sur le Web, en réponse à des requêtes http, des documents partiellement ou totalement dynamiques.

Origine :

Object Design

QORX

QORX n'est pas un SGBD, mais un petit utilitaire écrit en C++ (sources disponibles!) qui démontre comment il est possible de générer un document XML à partir d'une base de données disposant d'une interface ODBC, en réponse à une requête SQL.
Ce logiciel pouvant être copié et distribué librement sous réserve de conserver sa mention d'origine,

j'en ai fait une copie locale.
Cela ne doit pas vous empêcher d'aller visiter le site de Griffin Brown Digital Publishing qui nous offre ce logiciel.

Moteurs XLink/XPointer

PHYLIS

Implémentation expérimentale des liens XLink. Cela a l'air intéressant d'après la doc fournie, mais pour l'utiliser il faut programmer en VisualBasic. Je n'ai pas essayé.
Origine :
Eliot Kimber, Phylis.com

Hybrick

Le navigateur Hybrick de Fujitsu, decrit un peu plus loin, supporte les liens XLink et les pointeurs XPTR.
Origine :
Fujitsu

Moteurs XSL

XSLJ

Logiciel qui traduit une feuille de style XSL en feuille de style DSSSL. Conçu pour être utilisé avec le processeur DSSSL Jade qui génére le document formaté final. XSLJ génére un fichier DSSSL. Jade reprend ce fichier plus le document source XML et génère un fichier HTML, XML, ou RTF. Logiciel gratuit pour toute utilisation. Source disponible. Plateformes Un*x (Solaris, FreeBSD), et Windows 32 bits
Origine :
Henry S. Thompson, Language Technology Group of the Human Communica-tion Research Group, Univ. Edimburgh

MSXSL

Processeur XSL (version XSL-97) principalement dédié à la génération HTML. Un contrôle ActiveX permet l'inclusion des éléments générés dans une page HTML. Support incomplet des constructeurs DSSSL. Logiciel gratuit pour utilisation non commerciale. Source non disponible.
Origine :
Microsoft

Docproc

Processeur XSL écrit en Java, conçu pour être utilisé en servlet pour la génération à la volée de documents HTML à partir d'un source XML et d'une feuille de style XSL. Utilise le langage de script Pnuts à la place de ECMAScript.

Peut être utilisé sur toute machine avec une machine Java installée, mais pour une utilisation en servlet, le serveur doit être un serveur Java. Logiciel gratuit pour toute utilisation non commerciale. Source disponible.

Origine :

Sean Russell (Department of Physics, University of Oregon)

LotusXSL

Processeur XSL (nouvelle syntaxe 1999) utilisant le parseur XML for Java d'IBM. Ecrit en Java. Donne accès a l'API DOM.

Origine :

Lotus, distribué par IBM AlphaWorks

Editeurs

Xeena

Editeur écrit en Java, proposé par IBM. Très lent pour éditer de gros documents. En pratique inutilisable sauf pour des démos.

Origine :

IBM AlphaWorks.

XED

De son vrai nom "XML document instance editor", XED est un éditeur de documents bien formés. Il est non validant mais utilise la DTD lorsqu'elle existe pour offrir des racourcis clavier. Disponible pour les plateformes Windows 32 bits et Solaris 2.5.

Origine :

Henry S. Thompson, HCRC Language Technology Group, University of Edinburgh.

XML Notepad

Editeur structuré non validant. Il traite la DTD si elle existe lors de la lecture d'un document, mais autorise des modifications non valides. Nécessite impérativement d'avoir la version 4 ou 5 d'Internet Explorer installée sur le poste de travail, car l'éditeur utilise l'analyseur syntaxique XML inclus dans le butineur. Disponible pour plateforme Windows 32 bits (95, 98 ou NT).

Origine :

Microsoft

Visual XML

Editeur structuré non validant. Toute plateforme Java. License Version d'évaluation gratuite. Une version française (interface et documentation) est disponible.

Origine :

Pierre Morel, Quebec Inc.

XPose

Editeur structuré non validant. Toute plateforme Java. License Version d'évaluation gratuite. Une API Java documentée permet aux programmeurs Java d'étendre les fonctionnalités de l'éditeur.

Origine :

Intravenous Communication Inc.

SXML

"Major Mode" d'emacs pour l'édition XML. Editeur structuré validant incluant un parseur.

Origine :

BULL et INRIA

TDTD

Package de macros emacs pour l'édition XML ou SGML.

Origine :

Tony Graham (Mulberry Technologies, Inc.)

PSGML

PSGML est un "major mode" de GNU Emacs pour l'édition de documents SGML.

psgmlxml-19980218.zip
est un ensemble de "patches" donnant à PSGML la possibilité d'éditer des documents XML.

psgml-xpointer.el
est une fonction additionnelle à PSGML qui génère automatiquement un XPointer for any pour toute position dans un document XML or SGML.

Editeurs validants WYSIWYG

Adept Publisher

Editeur SGML professionnel, doté d'un mécanisme de formatage de qualité "typographie professionnelle". Sa dernière version permet de créer des documents XML. Attention toutefois: les documents XML générés ne sont pas accompagnés d'une feuille de style standardisée. Convient donc très bien pour des applications XML pour lesquels le produit visé est l'impression papier. L'intégration de XSL est prévue lorsque la norme sera adoptée. Version commerciale seulement.

Origine :

ArborText

Butineurs

Multidoc Pro

Mutltidoc Pro est un butineur SGML / XML utilisant les feuilles de style Synex Viewport. Il inclue un moteur Hytime, qui pour l'instant ne traite pas les liens dans leur syntaxe XLink. La version Multidoc Pro Publisher permet d'éditer les feuilles de style et les navigateurs. Le butineur affiche aussi bien les documents valides que les bien formés sans DTD. Version d'évaluation gratuite valable 30 jours, et version commerciale.

Origine :

CITEC

Hybrick

Hybrick est un visualisateur de documents SGML ou XML. Pour le formatage des documents, il utilise des feuilles de style écrites dans le langage DSSSL. Il exploite le moteur SGML / DSSSL Jade. Il est donc possible d'utiliser Hybrick pour visualiser des documents XML, après avoir généré la ou les feuilles de style DSSSL nécessaires, à l'aide de XSLJ. La version 0.8 (mars 1999) supporte en outre les liens XLink et les pointeurs XPTR.

Origine :

Fujitsu

Internet Explorer 5.0

La version 5 d'Internet Explorer est disponible depuis le 18 mars 1999. Elle permet à peu près de visualiser un document XML. A peu près seulement, car plusieurs bogues font que cette version n'est pas conforme à la norme XML 1.0. En particulier, le navigateur n'accepte pas de sections CDATA dans un document. Le support du codage de caractères iso-8859-1 est incorrect. La déclaration de ce codage ne permet malheureusement pas de visualiser correctement un document contenant des références numériques aux entités caractères du type é
Le formatage des objets XML peut éventuellement se faire sous contrôle d'une feuille de style CSS. Là encore, on peut regretter que le navigateur ne respecte pas parfaitement le standatd CSS-1, et a fortiori le standard CSS-2, pourtant adoptés avec le soutien de Microsoft!
XSL n'est pas disponible, ce que l'on ne peut pas reprocher à Microsoft, étant donné que cette norme est en gestation. Une autre limitation est beaucoup plus génante : IE5 ne donne pas accès au DOM, en tout cas, ne donne pas un accès conforme à l'API standardisée par le W3C. C'est une déception, car MS a depuis des mois annoncé urbi et orbi son soutien très fort à cette norme. Espérons que cela viendra très vite, dès la prochaine release mineure !
En résumé, pour une véritable utilisation opérationnelle XML, ce navigateur manque encore de maturité.

Origine :

Microsoft

Netscape Gecko

C'est le nom de code de la toute nouvelle technologie de Netscape, encore en développement. En

Janvier 1999, c'est encore inutilisable par l'utilisateur final, mais c'est un chantier à visiter pour les développeurs confirmés.

Origine :

Netscape

RDF

RDF for XML

Librairie Java qui implémente l'analyse et la sérialisation des déclarations RDF, et une mécanisme de requêtes. Ce n'est pas une application : il n'y pas d'interface utilisateur. En revanche, les programmeurs Java y trouveront les classes nécessaires pour construire une application. RDF for XML utilise le parseur « XML for Java » d'IBM.

Origine :

IBM AlphaWorks

SiRPAC

Le Simple RDF Parser & Compiler (SiRPAC) est un logiciel permettant de valider des déclarations RDF et d'obtenir la liste complète des triplets. Ce logiciel, développé par le W3C, est écrit en Java. Il peut être utilisé à travers le réseau ou être installé localement. En installation locale, il utilise n'importe quel parser XML disposant d'une interface SAX strictement conforme à la version 1.0 de cet API.
Origine :
W3C

SMIL

GRiNS

Produit par le CWI, le centre de recherche en informatique des Pays-Bas, GRiNS est une implémentation complète de la norme SMIL. Basé sur Python, SMIL est disponible pour les plateformes Windows 32 bits et Unix.

Origine :

CWI

RealPlayer G2

Visualisateur SMIL pouvant être utilisé en application autonome, ou en extension d'un butineur HTML

Origine :

RealNetworks

V-Active for RealSystem G2

Editeur de présentations SMIL qui intègre un certain nombre de codeurs de RealNetworks et permet à partir des fichiers bruts (.jpg, .avi, .txt) de générer des fichiers pour présentation au fil de l'eau dans les formats de RealNetworks. Il autorise un certain nombre d'opération de coupure et de montage sonore ou vidéo. La définition du layout se fait par édition graphique directe. Version d'évaluation gratuite, valable trente jours.

Origine :

Veon

Divers

Near&Far Designer

Editeur de DTD. L'utilisateur dispose d'une vue graphique de sa DTD. Il peut créer de nouveaux éléments et définir leur modèle de contenu uniquement avec des commandes par boutons et des boîtes de dialogue. Permet de créer une DTD sans risquer d'erreur syntaxique.

Origine:

Microstar

Jade

Processeur DSSSL. Prend en entrée un document XML ou SGML, et une feuille de style DSSSL. Permet de générer des documents formatés HTML, RTF, TeX.

Origine :

James Clark

Application XML hors W3C

-
- PGML
- VML
- CDF
- OSD
- OTP
- WEBDAV
- DCD
- BSML
- OMF
- XML-QL

PGML : Precision Graphics Markup Language

Objectifs : Définir un langage graphique 2D convenant aussi bien pour les petites illustrations graphiques des pages Web, qu'aux besoins des professionnels des arts graphiques. PGML est basé sur le même modèle graphique que PostScript et PDF. Il s'y ajoute :

Des possibilités d'animations, les plus simples étant obtenues par des primitives du langage et les plus complexes par une ouverture vers un langage de script.

La possibilité de nommer des objets et de définir des groupements nommés d'objets.

Des liens hypertextes (présents dans PDF mais pas dans PostScript).

La transparence des objets, qui permet de laisser apparaître un fond graphique ou un autre objet placé virtuellement “sous” un autre objet graphique.
Proposé par : groupe de travail associant Adobe, IBM, Netscape, Sun.
Référence : Nabeel Al-Shamma & al. Precision Graphics Markup Language (PGML). Note submitted to the World Wide Web Consortium, 10 Avril 1998, NOTE-PGML-19980410.

VML : Vector Markup Language

Objectifs : Définir un langage graphique 2D permettant d'insérer dans les documents HTML ou XML des graphiques structurés, supportant les changements d'échelle, les modifications par l'utilisateur (structure éditable), et les animations via un langage de script. Les objectifs fonctionnels sont donc à peu près les mêmes que ceux de PGML, mais l'architecture visée est sensiblement différente. En effet, PGML implique qu'un butineur devrait inclure, soit comme extension soit en interne dans son propre code, une « machine graphique PGML » qui serait très proche, à l'analyseur syntaxique près, des

machines graphiques PDF exis-tantes. VML, au contraire, est conçu pour exploiter les possibilités graphiques déjà dispo-nibles dans les butineurs HTML. Cet objectif se traduit par plusieurs choix qui différencient VML et PGML :

Il n'y a pas en VML d'objet graphique spécifique text. Bien sûr il est possible d'inclure des chaînes de caractères dans un groupe VML, mais ce texte sera rendu par la machine graphique CSS déjà disponible dans le butineur.

Le positionnement et la décoration (bordure, fond,...) des boîtes dans lesquelles vont être tracés les objets graphiques sont contrôlés par des propriétés CSS.

Les images bitmaps ne sont pas des objets spécifiques à VML : elles sont positionnées par rapport au texte dans une boite sous contrôle de propriétés CSS et sont rendues par le logiciel adéquat, comme n'importe quelle image incluse dans une page XML ou HTML.
VML ne définit qu'un seul objet graphique de base qui lui soit propre, la forme (shape), qui se spécialise en objets path qui présente à peu près les mêmes caractéristiques qu'un path PGML, et en formes (cercle,...) prédéfinies permettant un écriture plus concise. Bien entendu, les formes peuvent être associées en groupes, et les groupes peuvent s'emboîter dans d'autres groupes.
Proposé par : groupe de travail associant Autodesk, Hewlett-Packard, Macromedia, Microsoft, Visio
Référence : Brian Mathews & al. Vector Markup Language (VML). Note submitted to the World Wide Web Consortium, 13 May 1998. NOTE-VML-1998-0513.

CDF : Channel Definition Format

Objectifs : Rendre possible le « push » sur le Web c'est à dire permettre aux utilisateurs de s'abonner à des « chaînes d'information », et de recevoir automatiquement, lors de chaque connexion au réseau les mises à jour relatives à ces chaînes.
Proposé par : Microsoft
Référence : Castedo Ellerman. Channel Definition Format (CDF). Submission to the World Wide Web Consortium, 09 March 1997.

OSD : Open Software Description

Objectifs : Permettre la diffusion (vente, diffusion gratuite, mises à jour) par le réseau de logiciels et de progiciels. Plus précisément, OSD doit permettre :

Le téléchargement et l'installation automatique de logiciels sur le poste d'un utili-sa-teur, et ce quelque soit la plateforme (cpu, système d'exploitation) qu'il utilise.

La mise à jour de ce logiciel en mode “push”, c'est à dire sans imposer à l'utilisateur de devoir surveiller sur le site du fournisseurs les annonces de nouvelles versions.

L'installation automatique des logiciels en prenant en compte leurs dépendances à l'égard d'autres logiciels, bibliothèques, répertoires de classes, ou pilotes, et en ne téléchargeant que ce qui n'est pas déjà installé, dans la bonne version, sur le poste de l'utilisateur.
Proposé par : Microsoft, Marimba.
Référence : Arthur van Hoff, Hadi Partovi, Tom Thai. Specification for the Open Software

Description (OSD) Format. 11 Août 1997, Soumission au World Wide Web Consortium.

OTP : Internet Open Trading Protocol

Objectifs : Définition d'un protocole normalisé permettant l'échange de transactions financières et commerciales.
Proposé par : Consortium OTP, regroupant une trentaine de compagnies : banques, réseaux de cartes bancaires, technologies de l'information, commerce en ligne, etc.
Référence : The Open Trading Protocol Consortium. Internet Open Trading Protocol. Part 1 : Business Description. 12 January 1998. Part 2 : Specification. 12 January 1998.

WEBDAV : World Wide Web Distributed Authoring and Versioning

Objectifs : Conception des standards nécessaires (protocole, métadonnées) pour permettre l'édition coopérative et la gestion de versions de documents sur le Web. Cet objectif implique notamment les fonctionnalités suivantes :

Création et accès à des métadonnées permettant d'identifier les ressources, leur propriétaire, les droits d'accès afférents, les dates de modification, etc. Webdav ne définit pas un système de métadonnées particulier comme par exemple le Dublin-core (voir chapitre 7), mais un formalisme général de représentation et d'accès aux métadonnées. Celle-ci sont des paires attribut - valeur, attachées aux ressources qu'elles décrivent par des liens typés.

Permettre la copie ou le déplacement d'une ressource sans rendre immédiatement périmées les métadonnées qui la décrivent.

Obtenir selon une méthode standard la liste des ressources contenues dans un répertoire en ligne, et ce récursivement sur les sous-répertoires.

Mettre en place un mécanisme de préemption (verrouillage) lors de l'édition d'une ressource par un utilisateur autorisé, évitant qu'un autre utilisateur, lui aussi autorisé, puisse parallélement entreprendre l'édition de la même ressource, et lors de la sauvegarde de ses modifications puisse “écraser” le travail de son collègue.
Proposé par : IETF

Documents de référence :

WEBDAV Working Group, Extensions for Distributed Authoring on the World Wide Web - WEBDAV. Internet Draft <draft-ietf-webdav-protocol-08>.

DCD : Document Content Description for XML

Objectifs : DCD est la dernière en date de plusieurs propositions [XML-DATA, XSchema] visant à permettre la description en XML d'un schéma, c'est à dire d'une structure de données ou de documents XML, sans utiliser aucune des déclarations spéciales propres aux DTD. DCD est un formalisme basé sur RDF qui permet d'exprimer tout ce que l'on exprime ordinairement dans une DTD, et de plus, de typer les éléments et définir entre eux des relations conceptuelles.
Proposé par : Groupe de travail associant des représentants de Textuality, Microsoft et IBM.

Référence : Tim Bray, Charles Frankston, Ashok Malhotra. Document Content Description for XML. Submission to the World Wide Web Consortium, 31 July 1998, Note-dcd-19980731.

DDML : Document Definition Markup Language

Objectifs : He bien non, DCD n'est plus la dernière en date des propositions visant à permettre la description en XML d'un schéma. La dernière en date s'appelle DDML. C'est le résultat d'un travail cooperatif mené par des abonnés à la liste de diffusion XML-DEV.
Proposé par : Groupe de travail coordonné par Ronald Bouret et Simon St Laurent.
Référence : Ronald Bourret, John Cowan, Ingo Macherius, Simon St. Laurent, Document Definition Markup Language (DDML) Specification, Version 1.0. W3C Note, 19-Jan-1999, NOTE-ddml-19990119.

BSML : Bioinformatic Sequence Markup Language

Objectifs : Proptocole permettant l'échange et l'affichage de descriptions de molécules d'ADN, d'ARN et de séquences protéiniques. Le protocole, basé sur XML, permet de décrire des séquences d'un génome, et de leur attacher des propriétés graphiques en vue de leur affichage sur écran.
Proposé par : Visual Genomics Inc., projet TopoGEN, financé par le National Center for Human Genome Research.
Référence : Bioinformatic Sequence Markup Language (BSML): A Public Domain Protocol for Graphic Genomic Displays.

OMF : Weather Observation Markup Format

Objectifs : Définition d'une structure normalisée pour les bulletins météo permettant leur traitement automatique.
Proposé par : Armée US (Air Force, Navy)
Référence : Weather Observation Definition Format.

XMI : XML Metadata Interchange Format

Objectifs : Permettre l'échange de données (et métadonnées) entre les outils d'aide à la modélisation basés sur le langage UML (langage de modélisation de l'OMG), et entre ces outils et les répertoires de métadonnées conformes à l'architecture définie par MOF (Meta Object Facility).
Proposition de IBM, Unisys, Oracle, DSTC [Distributed Systems Technology Centre], et Platinum Technology à l'Open Management Group (OMG)
Référence : XML Metadata Interchange (XMI), Proposal to the OMG OA&DTF RFP 3: Stream-based Model Interchange Format (SMIF). OMG Document ad/98-07-01
July 6, 1998.

XML-QL : A Query Language for XML

Objectifs : Définir un langage de requète normalisé permettant l'accès à des fragments XML, c'est à dire en pratique, permettant d'interroger des bases de données XML. La proposition définit un modèle de graphe semi-structuré pour les données XML susceptibles d'être organisées selon des schémas (DTD) divers, et un langage de requète permettant la fusion de données.
Proposé par : AT&T, Univ. of Pennsylvania, Univ. of Washington, INRIA.
Référence :Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, Dan Suciu. XML-QL: A Query Language for XML, Submission to the World Wide Web Consortium 19-August-1998, NOTE-xml-ql-19980819

Seules figurent ici les normes de droit ou les standards industriels qui ont un rapport direct avec XML.

Recommandations du W3C

CSS 1	*Cascading Style Sheets, level 1*. W3C Recommendation, 17 December 1996, Reference REC-CSS1-961217.
CSS 2	*Cascading Style Sheets, level 2. CSS2 Specification*. W3C Recommendation, 12 May 1998, Reference REC-CSS2-19980512.
DOM	*Document Object Model (DOM) Level 1 Specification, Version 1.0*, W3C Recommendation. References: REC-DOM-Level-1-19981001
HTML 4.0	*HTML 4.0 Specification*. W3C Recommendation, revised on 24 Apr. 1998. Reference : REC-html40-19980424.
MathML	*Mathematical Markup Language (MathML) 1.0 Specification*, W3C Recommendation 07-April-1998. Reference : REC-MathML-19980407
PICS-label **PICS-Rating** **PICS-Rules**	Jim Miller, editor. *PICS Label Distribution Label Syntax and Communication Protocols*. W3C Recommendation 31-October-96 Jim Miller, editor. *Rating Services and Rating Systems (and Their Machine Readable Descriptions)*. W3C Recommendation. Martin Presler-Marshall, editor. *PICSRules 1.1*, W3C Recommendation 29 Dec. 1997
SMIL	*Synchronized Multimedia Integration Language (SMIL) 1.0 Specification*, W3C Recommendation 15 June 1998. Reference : REC-smil-19980615.
XML	*Extensible Markup Language (XML) 1.0.* W3C Recommendation 10 February 1998, Reference : REC-xml-19980210.
Namespaces	*Namespaces in XML.* W3C Recommendation 14 January 1999, Reference : REC-xml-names-19990114.

RDF-1	*Resource Description Framework (RDF) Model and Syntax Specification*. W3C W3C Recommendation 22 February 1999, Ref.: REC-rdf-syntax-19990222.
RDF-2	*Resource Description Framework (RDF) Schema Specification* , W3C Proposed Recommendation, 03 March 1999

Documents de travail pré-normatifs

StyleSheets	*Associating stylesheets with XML documents, Version 1.0.* W3C Proposed Recommendation 14 January 1999, Ref.: PR-xml-stylesheet-19990114.
XLink	*XML Linking Language (XLink)*. W3C Working-draft, 3 March 1998.
XPointer	*XML Pointer Language (Xpointer)*. W3C Working-draft, 3 March 1998.
XSL 97	Sharon Adler & al., *A Proposal for XSL*. Note submitted to the W3C on 27 August 1997.
XSL 98	*Extensible Stylesheet Language (XSL) Version 1.0*. References: WD-xsl-19981216, World Wide Web Consortium Working Draft, 16-December-1998

IETF

IANA	Internet Assigned Numbers Authority, *Official Names for Character Sets*, ed. Keld Simonsen & al.
RFC 1738 : URL	Internet Engineering Task Force (IETF). RFC 1738: *Uniform Resource Locators (URL)*, Ed. T. Berners-Lee, L. Masinter, M. McCahill. 1994.
RFC 1766	Internet Engineering Task Force (IETF). RFC 1766 : *Tags for the Identification of Languages*. Ed. H. Alvestrand, 1995.
RFC 1808 : URL	Internet Engineering Task Force (IETF). RFC 1808: *Relative Uniform Resource Locators*, Ed. R. Fielding. 1995.
RFC 2045 : MIME	Internet Engineering Task Force (IETF). RFC 2045 : Multipurpose Internet Mail Extensions (MIME) Part One : Format of Internet Message Bodies. ed. N. Freed & al. Nov. 1996.

RFC 2046 : MIME	Internet Engineering Task Force (IETF). RFC 2046 : Multipurpose Internet Mail Extensions (MIME) Part Two : Media Types. ed. N. Freed & al. Nov. 1996.
RFC 2068 : HTTP	Internet Engineering Task Force (IETF). RFC 2068 : Hypertext Transfer Protocol -- HTTP/1.1. ed. R. Fielding & al. Jan. 1997
RFC 2326	Internet Engineering Task Force (IETF). RFC 2326 : *Real Time Streaming Protocol (RTSP)*. ed. H. Schulzrinne & al. April 1998
RFC 2396 : URI	Internet Engineering Task Force (IETF). RFC 2396 : *Uniform Resource Identifiers (URI): Generic Syntax*. Ed : T. Berners-Lee & al. August 1998.

ISO et AFNOR

Les documents normatifs ne sont pas accessibles en ligne. Le catalogue des normes ISO dans le domaine des T.I. peut être consulté ici. Le catalogue de l'AFNOR est également accessible.

ISO 8879 (SGML)	ISO 8879:1986(E). Information processing -- Text and Office Systems -- *Standard Generalized Markup Language (SGML)*. First edition -- 1986-10-15. AFNOR : NF EN 28879 Décembre 1990 Traitement de l'information. Systèmes bureautiques. *Langage normalisé de balisage généralisé (SGML).*
ISO 10744 (HyTime)	ISO/IEC 10744-1992 (E). Information technology -- *Hypermedia/Time-based Structuring Language (HyTime).* *Extended Facilities Annexe*, 1996. AFNOR : NF ISO/CEI 10744 Décembre 1994 Technologies de l'information. *Langage de structuration hypermedia/événementiel (hytime).*
ISO-10179 (DSSSL)	ISO/IEC 10179:1996 Information technology -- Processing languages -- *Document Style Semantics and Specification Language (DSSSL).* AFNOR : NF ISO/CEI 10179 Décembre 1997 Technologies de l'information. Langages de traitement. *Sémantique de présentation de documents et langage de spécifications (DSSSL).*
ISO 639	ISO 639:1988 (E). *Code for the representation of names of languages*.

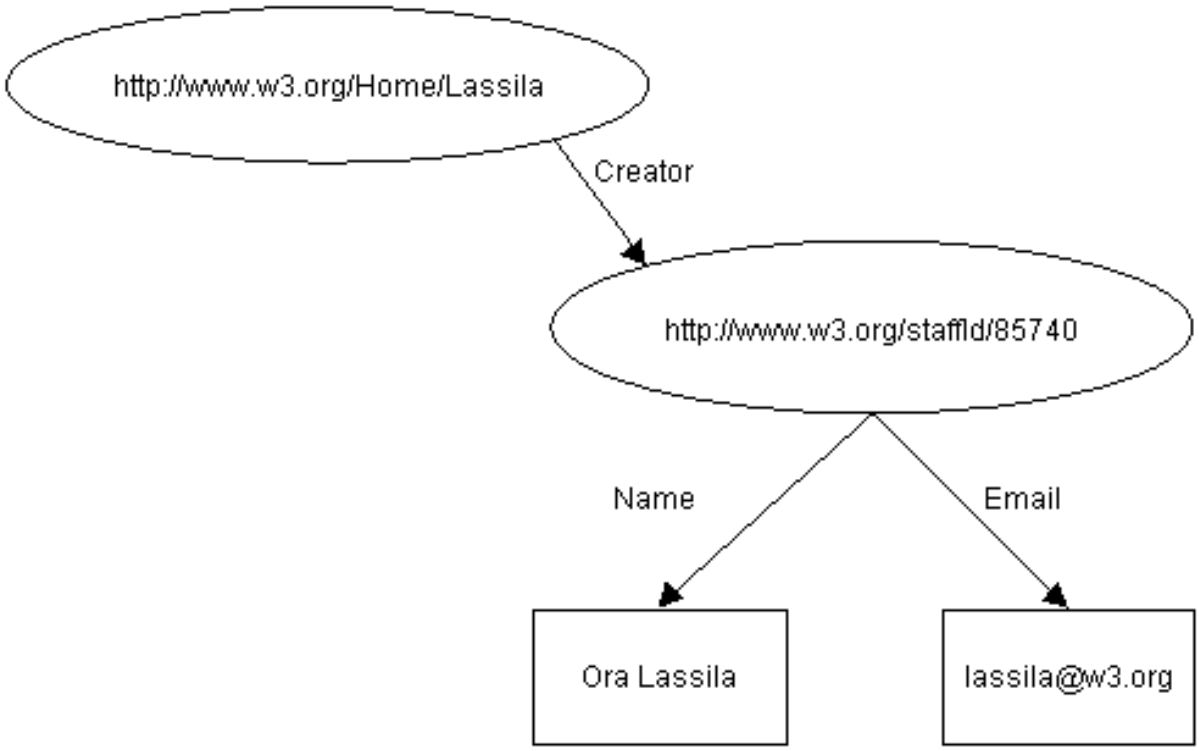
ISO 8601	ISO 8601:1988. Data elements and interchange formats - Information interchange - Representation od dates and times.
ISO 3166-1	ISO 3166-1:1997 (E). *Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes*. AFNOR : NF EN ISO 3166-1 Décembre 1997 *Codes pour la représentation des noms de pays et de leurs subdivisions. Partie 1 : codes pays.*
ISO/IEC 10646	ISO/IEC 10646-1993 (E). Information technology -- *Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane (plus amendments AM 1 through AM 7).*
ISO 2709 (MARC)	ISO 2709:1996 Information and documentation - *Format for Information Exchange*. AFNOR : NF ISO 2709 Décembre 1996 Information et documentation. *Format pour l'échange d'information.*
UTF-16	Amendment 1:1996 to ISO/IEC 10646-1:1993 *Transformation Format for 16 planes of group 00 (UTF-16)*. ISO, Geneva, 1996.
UTF-8	Amendment 2:1996 to ISO/IEC 10646-1:1993 UCS *Transformation Format 8 (UTF-8)*. ISO, Geneva, 1996.

Divers

ECMAScript	Standard ECMA-262 *ECMAScript: A general purpose, cross-platform programming language* (June 1997). Copie locale (RTF.ZIP).
SMPTE	Society of Motion Picture & Television Engineers. *Time and Control Codes for 24, 25 or 30 Frame-per-Second Motion-Picture Systems*. RP 136-1995.

Exemple

- La personne référencée par le numéro d'employée "85740" se nome "Ora Lassila" et a un email "lassila@w3.org".
- La ressource "http://www.w3.org/Home/Lassila" a été crée par cette personne.



s'écrit en RDF/XML en forme sérialisée

```
<rdf:RDF>
   <rdf:Description about="http://www.w3.org/Home/Lassila">
       <s:Creator rdf:resource="http://www.w3.org/staffId/85740"/>
   </rdf:Description>

    <rdf:Description about="http://www.w3.org/staffId/85740">
        <v:Name>Ora Lassila</v:Name>
        <v:Email>lassila@w3.org</v:Email>
    </rdf:Description>
</rdf:RDF>
```

Il a donc deux ressources décrites mais il n'est pas évident pour le lecteur que la deuxième ressource utilisée est a l'intérieur de la première description.

La même expression peut s'écrire de la façon suivante ce qui est plus lisible pour le lecteur

```
<rdf:RDF>
 <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>
    <rdf:Description about="http://www.w3.org/staffId/85740">
       <v:Name>Ora Lassila</v:Name>
       <v:Email>lassila@w3.org</v:Email>
    </rdf:Description>
```

```
    </s:Creator>
 </rdf:Description>
</rdf:RDF>
```

La même expression peut s'écrire de la façon suivante plus compacte

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
  <s:Creator rdf:resource="http://www.w3.org/staffId/85740"
             v:Name="Ora Lassila"
             v:Email="lassila@w3.org" />
</rdf:Description>
</rdf:RDF>
```

Example: Dublin Core Metadata:

Propriétés descriptives de données documentaires (Title, Author, Subject, Description, Publisher, Date, Format, Language, etc.)

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.0/">
  <rdf:Description about="http://www.w3.org">
    <dc:Publisher>World Wide Web Consortium</s:Publisher>
    <dc:Title>W3C Home Page</s:Title>
    <dc:Date>1998-10-03T02:27</s:Date>
  </rdf:Description>
</rdf:RDF>
```

équivalent a celui ci

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.0/">
  <rdf:Description about="http://www.w3.org"
       dc:Publisher="World Wide Web Consortium"
       dc:Title="W3C Home Page"
       dc:Date="1998-10-03T02:27"/>
</rdf:RDF>
```

Valider le RDF ci-dessus avec SiRPAC; un RDF Parser and Compiler, ecrit by Janne Saarela (W3C).

Autre exemple utilisant les propriétés du Dublin Core :

```
  <rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/metadata/dublin_core#">
    <rdf:Description about="http://www.dlib.org">
      <dc:Title>D-Lib Program - Research in Digital Libraries</dc:Title>
      <dc:Description>The D-Lib program supports the community of people
       with research interests in digital libraries and electronic
       publishing.</dc:Description>
      <dc:Publisher>Corporation For National Research Initiatives</dc:Publisher>
      <dc:Date>1995-01-07</dc:Date>
      <dc:Subject>
        <rdf:Bag>
          <rdf:li>Research; statistical methods</rdf:li>
          <rdf:li>Education, research, related topics</rdf:li>
          <rdf:li>Library use Studies</rdf:li>
        </rdf:Bag>
      </dc:Subject>
      <dc:Type>World Wide Web Home Page</dc:Type>
      <dc:Format>text/html</dc:Format>
      <dc:Language>en</dc:Language>
    </rdf:Description>
  </rdf:RDF>
```

Example

Dublin Core Metadata:

Dublin Core Schema

Ce shema décrit la sémantique des propriétés définies par le Dublin Core.

```
<?xml version='1.0'?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
   xmlns:dc="">

<rdf:Description about = "">
  <dc:Title> The Dublin Core Element Set </dc:Title>
  <dc:Creator> The Dublin Core Metadata Inititative </dc:Creator>
  <dc:Description> The Dublin Core is a simple metadata element
      set intended to facilitate discovery of electronic
      resources. </dc:Description>
  <dc:Date> 1995-03-01 </dc:Date>
</rdf:Description>

<rdf:Description ID="Title">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Title</rdfs:label>
  <rdfs:comment>The name given to the resource, usually by the Creator
  or Publisher.</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource = ""/>
</rdf:Description>

<rdf:Description ID="Creator">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Author/Creator</rdfs:label>
  <rdfs:comment>The person or organization primarily responsible for
  creating the intellectual content of the resource. For example,
  authors in the case of written documents, artists, photographers, or
  illustrators in the case of visual resources.</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource = ""/>
</rdf:Description>

<rdf:Description ID="Subject">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Subject</rdfs:label>
  <rdfs:comment>The topic of the resource. Typically, subject will be
  expressed as keywords or phrases that describe the subject or
  content of the resource. The use of controlled vocabularies and
  formal classification schemes is encouraged.</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource = ""/>
</rdf:Description>

<rdf:Description ID="Description">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Description</rdfs:label>
  <rdfs:comment> A textual description of the content of the resource,
  including abstracts in the case of document-like objects or content
  descriptions in the case of visual resources.</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource = ""/>
</rdf:Description>
```

```
<rdf:Description ID="Publisher">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Publisher</rdfs:label>
  <rdfs:comment>The entity responsible for making the resource
  available in its present form, such as a publishing house, a
  university department, or a corporate entity.</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource = ""/>
</rdf:Description>

<rdf:Description ID="Contributor">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Other Contributors</rdfs:label>
  <rdfs:comment>A person or organization not specified in a Creator
  element who has made significant intellectual contributions to the
  resource but whose contribution is secondary to any person or
  organization specified in a Creator element (for example, editor,
  transcriber, and illustrator).</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource = ""/>
</rdf:Description>

<rdf:Description ID="Date">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Date</rdfs:label>
  <rdfs:comment>A date associated with the creation or availability of
  the resource. Such a date is not to be confused with one belonging
  in the Coverage element, which would be associated with the resource
  only insofar as the intellectual content is somehow about that
  date. Recommended best practice is defined in a profile of ISO 8601
  [Date and Time Formats (based on ISO8601), W3C Technical Note,
  http://www.w3.org/TR/NOTE-datetime] that includes (among others)
  dates of the forms YYYY and YYYY-MM-DD. In this scheme, for example,
  the date 1994-11-05 corresponds to November 5, 1994.</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource = ""/>
</rdf:Description>

<rdf:Description ID="Type">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Type</rdfs:label>
  <rdfs:comment>The category of the resource, such as home page,
  novel, poem, working paper, technical report, essay, dictionary. For
  the sake of interoperability, Type should be selected from an
  enumerated list that is currently under development in the workshop
  series.</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource = ""/>
</rdf:Description>

<rdf:Description ID="Format">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Format</rdfs:label>
  <rdfs:comment>The data format of the resource, used to identify the
  software and possibly hardware that might be needed to display or
  operate the resource. For the sake of interoperability, Format
  should be selected from an enumerated list that is currently under
  development in the workshop series.</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource = ""/>
</rdf:Description>

<rdf:Description ID="Identifier">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Identifier</rdfs:label>
```

```
    <rdfs:comment>A string or number used to uniquely identify the
    resource. Examples for networked resources include URLs and URNs
    (when implemented). Other globally-unique identifiers, such as
    International Standard Book Numbers (ISBN) or other formal names are
    also candidates for this element.</rdfs:comment>
    <rdfs:isDefinedBy rdf:resource = ""/>
  </rdf:Description>

  <rdf:Description ID="Source">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:label>Source</rdfs:label>
    <rdfs:comment>Information about a second resource from which the
    present resource is derived. While it is generally recommended that
    elements contain information about the present resource only, this
    element may contain a date, creator, format, identifier, or other
    metadata for the second resource when it is considered important for
    discovery of the present resource; recommended best practice is to
    use the Relation element instead.  For example, it is possible to
    use a Source date of 1603 in a description of a 1996 film adaptation
    of a Shakespearean play, but it is preferred instead to use Relation
    "IsBasedOn" with a reference to a separate resource whose
    description contains a Date of 1603. Source is not applicable if the
    present resource is in its original form.</rdfs:comment>
    <rdfs:isDefinedBy rdf:resource = ""/>
  </rdf:Description>

  <rdf:Description ID="Language">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:label>Language</rdfs:label>
    <rdfs:comment>The language of the intellectual content of the
    resource. Where practical, the content of this field should coincide
    with RFC 1766 [Tags for the Identification of Languages,
    http://ds.internic.net/rfc/rfc1766.txt ]; examples include en, de,
    es, fi, fr, ja, th, and zh.</rdfs:comment>
    <rdfs:isDefinedBy rdf:resource = ""/>
  </rdf:Description>

  <rdf:Description ID="Relation">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:label>Relation</rdfs:label>
    <rdfs:comment>An identifier of a second resource and its
    relationship to the present resource. This element permits links
    between related resources and resource descriptions to be
    indicated. Examples include an edition of a work (IsVersionOf), a
    translation of a work (IsBasedOn), a chapter of a book (IsPartOf),
    and a mechanical transformation of a dataset into an image
    (IsFormatOf). For the sake of interoperability, relationships should
    be selected from an enumerated list that is currently under
    development in the workshop series.</rdfs:comment>
    <rdfs:isDefinedBy rdf:resource = ""/>
  </rdf:Description>

  <rdf:Description ID="Coverage">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:label>Coverage</rdfs:label>
    <rdfs:comment>The spatial or temporal characteristics of the
    intellectual content of the resource. Spatial coverage refers to a
    physical region (e.g., celestial sector); use coordinates (e.g.,
    longitude and latitude) or place names that are from a controlled
    list or are fully spelled out. Temporal coverage refers to what the
```

```
    resource is about rather than when it was created or made available
    (the latter belonging in the Date element); use the same date/time
    format (often a range) [Date and Time Formats (based on ISO8601),
    W3C Technical Note, http://www.w3.org/TR/NOTE-datetime] as
    recommended for the Date element or time periods that are from a
    controlled list or are fully spelled out.</rdfs:comment>
    <rdfs:isDefinedBy rdf:resource = ""/>
  </rdf:Description>

  <rdf:Description ID="Rights">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:label>Rights</rdfs:label>
    <rdfs:comment>A rights management statement, an identifier that
    links to a rights management statement, or an identifier that links
    to a service providing information about rights management for the
    resource.</rdfs:comment>
    <rdfs:isDefinedBy rdf:resource = ""/>
  </rdf:Description>
  </rdf:RDF>
```

Exemple de metadata et SMIL

Meta-data inclus dans un document SMIL.

Cet exemple utilise le Schema du Dublin Core version 1.0 [[DC]] et le SMIL Metadata Schema:

```
<?xml version="1.0" ?>
<smil xmlns = "http://www.w3.org/TR/.../SMIL-Boston.dtd">
 <head>
 <meta id="meta-smil1.0-a" name="Publisher" content="W3C" />
 <meta id="meta-smil1.0-b" name="Date" content="1999-10-12" />
 <meta id="meta-smil1.0-c" name="Rights" content="Copyright 1999 John Smith" />
 <meta id="meta-smil1.0-d" http-equiv="Expires" content=" 31 Dec 2001 12:00:00 GMT">


  <metadata id="meta-rdf">
   <rdf:RDF
       xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:rdfs = "http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
       xmlns:dc = "http://purl.org/metadata/dublin_core#"
       xmlns:smilmetadata = "http://www.w3.org/AudioVideo/.../smil-ns#" >

<!-- Metadata about the SMIL presentation -->
   <rdf:Description about="http://www.foo.com/meta.smi"
       dc:Title="An Introduction to the Resource Description Framework"
       dc:Description="The Resource Description Framework (RDF) enables the encoding,
exchange and reuse of structured metadata"
       dc:Publisher="W3C"
       dc:Date="1999-10-12"
       dc:Rights="Copyright 1999 John Smith"
       dc:Format="text/smil" >
       <dc:Creator>
          <rdf:Seq ID="CreatorsAlphabeticalBySurname">
             <rdf:li>Mary Andrew</rdf:li>
             <rdf:li>Jacky Crystal</rdf:li>
          </rdf:Seq>
       </dc:Creator>
       <smilmetadata:ListOfVideoUsed>
          <rdf:Seq ID="VideoAlphabeticalByFormatname">
              <rdf:li Resource="http://www.foo.com/videos/meta-1999.mpg"/>
              <rdf:li Resource="http://www.foo.com/videos/meta2-1999.mpg"/>
          </rdf:Seq>
       </smilmetadata:ListOfVideoUsed>
       <smilmetadata:Access LevelAccessibilityGuidelines="AAA"/>
   </rdf:Description>

<!-- Metadata about the video -->
   <rdf:Description about="http://www.foo.com/videos/meta-1999.mpg"
        dc:Title="RDF part one"
        dc:Creator="John Smith"
        dc:Subject="Metadata,RDF"
        dc:Description="RDF basic fonctionalities"
        dc:Publisher="W3C Press Service"
        dc:Format="video/mpg"
        dc:Language="en"
        dc:Date="1999-10-12"
        smilmetadata:Duration="60 secs"
        smilmetadata:VideoCodec="MPEG2" >
        <smilmetadata:ContainsSequences>
           <rdf:Seq ID="ChronologicalSequences">
```

```
            <rdf:li Resource="http://www.foo.com/videos/meta-1999.mpg#scene1"/>
            <rdf:li Resource="http://www.foo.com/videos/meta-1999.mpg#scene2"/>
          </rdf:Seq>
        </smilmetadata:ContainsSequences>
   </rdf:Description>

<!-- Metadata about a scene of the video -->
   <rdf:Description about="#scene1"
        dc:Title="RDF intro"
        dc:Description="Introduction to RDF fonctionalities"
        dc:Language="en"
        smilmetadata:Duration="30 secs"
        smilmetadata:Presenter="David Jones" >
        <smilmetadata:ContainsShots>
           <rdf:Seq ID="ChronologicalShots">
             <rdf:li>Panorama-shot</rdf:li>
             <rdf:li>Closeup-shot</rdf:li>
           </rdf:Seq>
        </smilmetadata:ContainsShots>
   </rdf:Description>
  </rdf:RDF>
 </metadata>

<!-- SMIL presentation -->
 <layout>
    <region id="a" top="5" />
 </layout>
 </head>
 <body>
 <seq>
   <video region="a" src="/videos/meta-1999.mpg" >
     <area id="scene1" begin="0" end ="30"/>
     <area id="scene2" begin="30" end ="60"/>
   </video>
   <video region="a" src="/videos/meta2-1999.mpg"/>
 </seq>
 </body>
</smil>
```

Note: Validate the above RDF description with SiRPAC; a Simple RDF Parser and Compiler, written by Janne Saarela (W3C).

Exemple: Micropayment

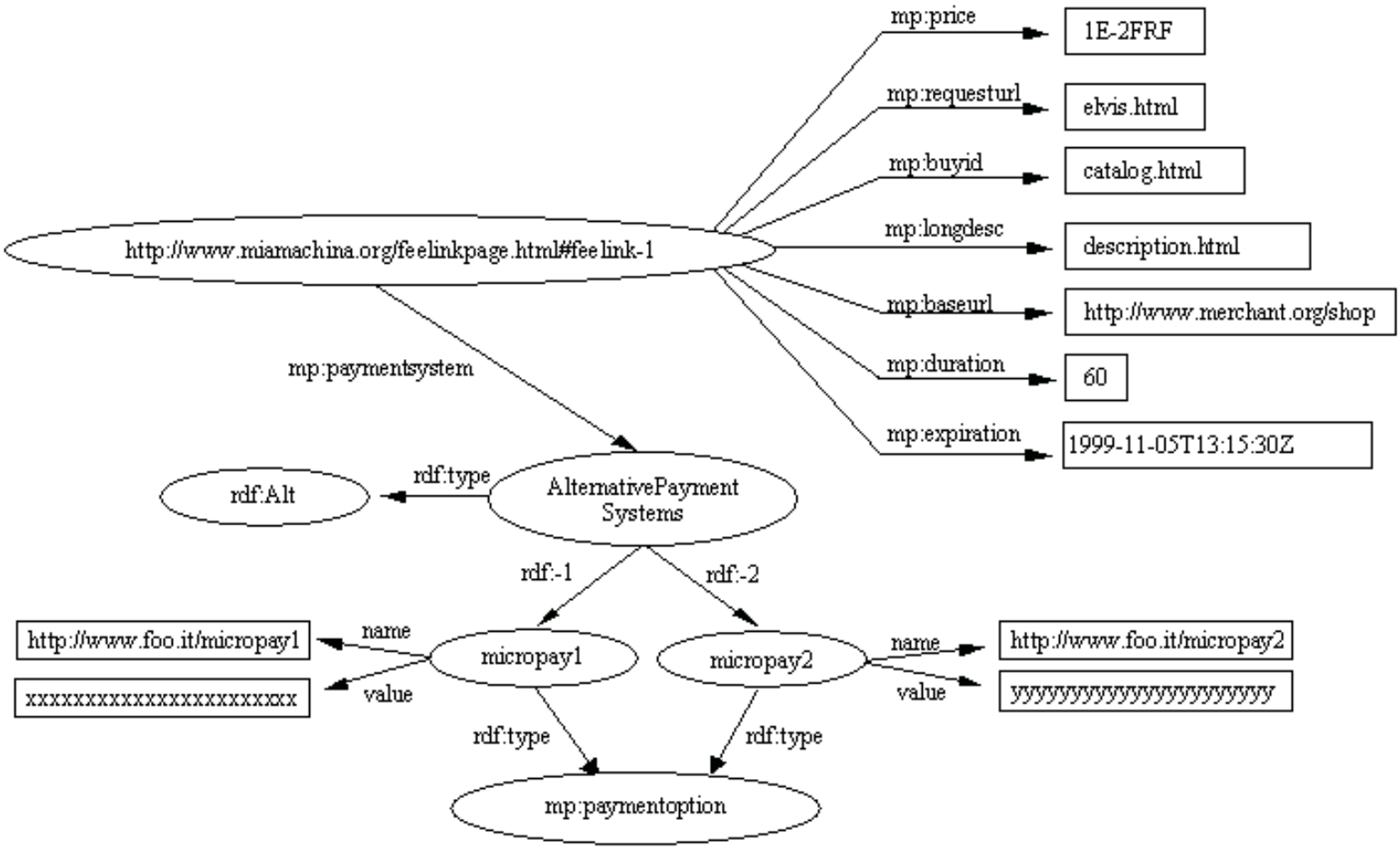
Fichier HTML

```
<HTML>
<HEAD>
<TITLE>Example of Fee Link </TITLE>
<LINK rel="meta" href="Micropayment-links.RDF">
</HEAD>

<BODY>
<A href="elvis.html" id="feelink-1"
title="Biography of Elvis">
<IMG src="http://www.merchant.org/product.gif"
     alt="Buy the Biography of Elvis through this Per-fee-link"> </A>
</BODY>
</HTML>
```

Fichier RDF

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:mp="http://www.w3.org/schema/micropay#">
  <rdf:Description about="http://www.biblio/elvis.html#feelink-1"
    mp:price="1E-2FRF"
    mp:duration="60"
    mp:expiration="1999-11-05T13:15:30Z">
    <mp:baseurl rdf:resource="http://www.merchant.org/shop"/>
    <mp:buyid rdf:resource="catalog.html"/>
    <mp:requesturl rdf:resource="elvis.html"/>
    <mp:longdesc rdf:resource="description.html"/>
    <mp:paymentsystem>s
      <rdf:Alt>
        <rdf:li>
          <mp:paymentoption rdf:ID="micropay1" name="http://www.foo.it/micropay1"
          rdf:value="xxxxxxxxxxxxxxxxx"/> </rdf:li>
        <rdf:li>
          <mp:paymentoption rdf:ID="micropay2" name="http://www.foo.it/micropay2"
          rdf:value="yyyyyyyyyyyyyyyyy"/> </rdf:li>
      </rdf:Alt>
    </mp:paymentsystem>
  </rdf:Description>
</rdf:RDF>
```



Valider le RDF ci-dessus avec SiRPAC; un RDF Parser and Compiler, ecrit by Janne Saarela (W3C).

Exemple MathML d'une équation :

$x^2 + 4x + 4 = 0$

Représentation par des balises de présentation

Les balises de présentation commencent par "m" et utilisent "o" pour operator "i" pour identifier "n" pour nombre, etc.
Les balises "mrow" groupes des objets horizontalement.

```
<math>
<mrow>
  <mrow>
        <msup> <mi>x</mi> <mn>2</mn> </msup> <mo>+</mo>
            <mrow>
              <mn>4</mn>
              <mo>&invisibletimes;</mo>
              <mi>x</mi>
            </mrow>
    <mo>+</mo>
    <mn>4</mn>
  </mrow>
    <mo>=</mo>
    <mn>0</mn>
</mrow>
</math>
```

Exemple: Représentation par des balises de sémantique

(concepts comme "times" "power of" etc).

$x^2 + 4x + 4 = 0$

```
    <apply>
        <plus/>
        <apply>
            <power/>
            <ci>x</ci>
            <cn>2</cn>
        </apply>
        <apply>
            <times/>
            <cn>4</cn>
            <ci>x</ci>
        </apply>
       <cn>4</cn>
    </apply>
```

Exemple SVG: deux groupes comprenant deux rectangles et de couleur differntes

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
  "http://www.w3.org/TR/2000/03/WD-SVG-20000303/DTD/svg-20000303-stylable.dtd">
<svg width="4in" height="3in">
  <desc>Two groups, each of two rectangles
  </desc>
  <g style="fill:red">
    <rect x="100" y="100" width="100" height="100" />
    <rect x="300" y="100" width="100" height="100" />
  </g>
  <g style="fill:blue">
    <rect x="100" y="300" width="100" height="100" />
    <rect x="300" y="300" width="100" height="100" />
  </g>
</svg>
```

Voir cet exemple SVG

Exemple SVG: deux groupes comprenant deux elipses couleur differentes

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
  "http://www.w3.org/TR/2000/03/WD-SVG-20000303/DTD/svg-20000303-stylable.dtd">
<svg width="12cm" height="4cm" viewBox="0 0 1200 400">
  <desc>Example ellipse01 - ellipses expressed in user coordinates</desc>

  <g transform="translate(300 200)">
    <ellipse rx="250" ry="100"
          style="fill:red" />
  </g>

  <ellipse transform="translate(900 200); rotate(30)"
        rx="250" ry="100"
        style="fill:none; stroke:blue; stroke-width: 20" />

</svg>
```

Voir cet exemple SVG

Exemple SVG: deux polygones

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
  "http://www.w3.org/TR/2000/03/WD-SVG-20000303/DTD/svg-20000303-stylable.dtd">
<svg width="12cm" height="4cm" viewBox="0 0 1200 400">
  <desc>Example polygon01 - star and hexagon</desc>

  <polygon style="fill:red; stroke:blue; stroke-width:10"
           points="350,75  379,161 469,161 397,215
                   423,301 350,250 277,301 303,215
                   231,161 321,161" />
  <polygon style="fill:lime; stroke:blue; stroke-width:10"
           points="850,75  958,137.5 958,262.5
                   850,325 742,262.6 742,137.5" />

</svg>
```

Voir cet exemple SVG

Insertion de SVG dans un document XML

```
 <?xml version="1.0" standalone="yes"?>
<parent xmlns="http://someplace.org"
       xmlns:svg="http://www.w3.org/2000/svg-20000303-stylable">
   <!-- du code parent ici -->
   <svg:svg width="5cm" height="8cm">
      <svg:ellipse rx="200" ry="130" />
   </svg:svg>
   <!-- du code parent ici -->
</parent>
</svg>
```

Exemple de Transition

```
 <smil>
    <head>
        <layout>
            <root-layout width="256" height="256" background-color="#000000"/>
            <region id="whole" left="32" top="32" width="192" height="192"/>
        </layout>
        <transition id="wipe1" type="wipe" subtype="slideHorizontal" dur="1s"/>
        <transition id="xfade1s" type="fade" subtype="crossfade"     dur="1s"/>

    </head>
    <body>
        <seq>
            <img src="butterfly.jpg" region="whole" dur="5s"/>
            <img src="eagle.jpg"     region="whole" dur="5s" transition="wipe1"/>
            <img src="wolf.jpg"      region="whole" dur="5s" transition="xfade1s"/>
            <img src="seal.jpg"      region="whole" dur="5s" transition="wipe1"/>
        </seq>
    </body>
</smil>
```

Exemple d'Animation

```
< ...>
<animate targetElement="foo" attributeName="width" from="10px" to="100px"
begin="0s" dur="10s" />
<animate targetElement="foo" attributeName="height" from="100px" to="10px"
begin="0s" dur="10s" />

< ...>
```

SMIL: différents types de liens

Exemple 1

la traversée du lien lance la nouvelle présentation qui remplace la présentation qui s'exécutait.

```
<a href="http://www.cwi.nl/somewhereelse.smi">
     <video src="rtsp://foo.com/graph.imf" region="l_window"/>
</a>
```

Exemple 2

la traversée du lien lance la nouvelle présentation qui s'ajoute a la présentation qui s'exécutait.

```
<a href="http://www.cwi.nl/somewhereelse.smi" show="new">
     <video src="rtsp://foo.com/graph.imf" region="l_window"/>
</a>
```

Exemple 3 : utilisant <anchor>

1) Associer des links a des coordonnées spatiales

```
<video src="http://www.w3.org/CoolStuff">
  <anchor href="http://www.w3.org/AudioVideo" coords="0%,0%,50%,50%"/>
  <anchor href="http://www.w3.org/Style"      coords="50%,50%,100%,100%"/>
</video>
```

2) Associer des links a des sous ensembles temporels

```
<video src="http://www.w3.org/CoolStuff">
  <anchor href="http://www.w3.org/AudioVideo" begin="0s" end="5s"/>
  <anchor href="http://www.w3.org/Style"      begin="5s" end="10s"/>
</video>
```

3) Aller a un sous-ensemble d'un media

```
Presentation A:

<a href="http://www.cwi.nl/mm/presentationB#tim">
   <video id="graph" src="rtsp://foo.com/graph.imf" region="l_window"/>
</a>


Presentation B:(www.cwi.nl/mm/presentationB)

<video src="http://www.w3.org/CoolStuff">
  <anchor id="joe" begin="0s" end="5s"/>
  <anchor id="tim" begin="5s" end="10s"/>
</video>
```

4) Combiner différentes utilisations de liens

```
Presentation A:
<smil>
<-- some stuff -->
```

```
<a href="http://www.cwi.nl/mm/presentationB#tim">
  <video id="graph" src="rtsp://foo.com/graph.imf" region="l_window"/>
</a>


Presentation B:
<smil>
<-- some stuff -->
<video src="http://www.w3.org/CoolStuff">
  <anchor id="joe" begin="0s" end="5s" coords="0%,0%,50%,50%"
          href="http://www.w3.org/"/>
  <anchor id="tim" begin="5s" end="10s" coords="0%,0%,50%,50%"
          href="http://www.w3.org/Tim"/>
</video>
```

SMIL: controle de contenu (switch)

```
<?xml version="1.0"
<smil>
   <head>
     <layout>
       <region id="video" top="5" left"15" height="100" width="300"/>
     </layout>
   </head>
   <body>
       <par>
          <video src="joe-video" region=video/>
          <switch>
              <audio src="joe-audio-french" system-language="fr"/>
              <audio src="joe-audio-french" system-language="it"/>
              <audio src="joe-audio-english"/>
          </switch>
       </par>
   </body>
</smil>
```

```
Bande passante:
.....
<switch>

<video src="high-quality-movie.rm" system-bitrate="40000"/>
<video src="medium-quality-movie.rm" system-bitrate="24000"/>
<video src="low-quality-movie.rm" system-bitrate="10000"/>
</switch>
.....
```

```
Nombre de couleurs:
...
<par>
  <text .../>
  <switch>
    <par system-screen-size="1280X1024"
         system-screen-depth="16">
    ........
    </par>

    <par system-screen-size="640X480"
         system-screen-depth="32">
    ...

    </par>
```

```
    <par system-screen-size="640X480"
         system-screen-depth="16">
    ...
    </par>
  </switch>
</par>

...

Soustritrage:

...

<seq>
  <par>
    <audio      src="audio.rm"/>
    <video      src="video.rm"/>
    <textstream src="closed-caps.rtx"
     system-captions="on"/>
  </par>
</seq>
...
```

Exemple: Structure de base SMIL

```
<?xml version="1.0"
<smil>
   <head>
     <layout>
       <root-layout  height="1300" width="600"/>
       <region id="video" top="5" left"15" height="100" width="300"/>
       <region id="image" top="115" left"15" height="10" width="300"/>
     </layout>
   </head>
   <body>
       <par>
          <video src="joe-video" region=video/>
          <audio src="joe-audio"/>
          <img src="copyrights-joe-video" region=image/>
        </par>
   </body>
</smil>
```

Example d'un document XHTML 1.1:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" >
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <p>Moved to <a href="http://vlib.org/">vlib.org</a>.</p>
    <hr/>
 </body>
</html>
```

Example simple d'un document XHTML 1.0:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <p>Moved to <br/>
    <a href="http://vlib.org/">vlib.org</a>.</p>
<hr/>
  </body>
```

Example d'integration de MathML dans un document XHTML 1.0:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" >
  <head>
    <title>A Math Example</title>
  </head>
  <body>
    <p>The following is MathML markup:</p>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply> <log/>
        <logbase>
          <cn> 3 </cn>
        </logbase>
        <ci> x </ci>
      </apply>
    </math>
  </body>
</html>
```

Example d'integration de XHTML 1.0 dans un autre XML namespace:

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns='http://www.w3.org/2000/02/book'
      xmlns:isbn="http://foo.org/isbn"/>
  <title>Cheaper by the Dozen</title>
  <isbn:number>1568491379</isbn:number>
  <notes>
    <!-- make HTML the default namespace for a hypertext commentary -->
    <p xmlns='http://www.w3.org/1999/xhtml'>
        This is also available <a href="http://www.w3.org/">online</a>.
    </p>
  </notes>
</book>
```

XForms: Typage des données

String

Facet	Description	Default
`min`	minimum length in characters	0
`max`	maximum length in characters	unlimited
`mask`	simple mask, e.g. "`ddd-ddd-dddd`"	no restriction
`pattern`	regular expression, e.g. "`\d{3}-\d{3}-\d{4}`"	no restriction

Basic Patterns (Masks)

Character Class	Description	Equiv. Regex	Default Representation
`letter`	All letter characters	`\p{L}`	'l'
`digit`	All digit characters	`\d`	'd'
`character`	All characters allowed in XML names	`\c`	'c'
`space`	All whitespace characters	`\s`	's'
`any`	Any Unicode character except newline	`.` (dot)	'.'

Différents type de pattern:

```
<string name="postalcode">
  <mask>ddddd</mask>        <!-- US ZIP code -->
  <mask>ddddd-dddd</mask>   <!-- US ZIP+4 code -->
  <mask>lldsdll</mask>      <!-- UK postal code -->
  <mask>llddsdll</mask>     <!-- UK postal code -->
  <mask>ldlsdld</mask>      <!-- Canadian postal code -->
</string>
```

Nombres

Facet	Description	Default
`min`	minimum value	minus infinity
`max`	maximum value	plus infinity
`integer`	if *"true"* only integer values are permitted	real numbers
`decimals`	how many digits after the decimal point are significant	unlimited

Exemple d'un entier non nul:

```
<number name="count" min="1" integer="true"/>
```

Valeurs monétaires

Facet	Description	Default
`min`	minimum value, e.g. "0"	minus infinity
`max`	maximum value	plus infinity
`decimals`	how many digits after the decimal point are significant	unlimited
`currency`	a space separated list of 3 letter currency codes, e.g. `USD` or `GBP`	unspecified

Exemple d'une valeur non-négative en British Pounds:

```
<money name="price" currency="GBP" decimals="2" min="0"/>
```

Dates

Les Dates sont définies en années, mois, jours [ISO 8601] standard .

Facet	Description	Default
`min`	minimum value or "`now`"	the distant past
`max`	maximum value or "`now`"	the distant future
`precision`	"`years`", "`months`" or "`days`"	unconstrained

Exemple d'une date de naissance:

```
<date name="birth" max="now"/>
```

Exemple d'une date d'expiration de carte de crédit:

```
<date name="expires" precision="months" min="now" max="+P4Y"/>
```

Valeurs calculées

le formulaire inclue des valeurs qui sont calculées depuis d'autres valeurs de champs:

Exemple:

```
<currency name="totalPrice" calc="sum(lineItem, quantity * price)"/>
```

Cet exemple somme le produit des valeurs `quantity` et `price` avec `lineItem`.

Énumérations/Union

```
<union name="weekday">
  <string range="closed">
    <value>Monday</value>
```

```
    <value>Tuesday</value>
    <value>Wednesday</value>
    <value>Thursday</value>
    <value>Friday</value>
    <value>Saturday</value>
    <value>Sunday</value>
  </string>
  <number min="1" max="7" integer="true"/>
</union>
```

Exemples valides:

```
<weekday>Tuesday</weekday>
<weekday>2</weekday>
```

XForms 1.0: Exemple d'utilisation

Exemple: ordre d'achat incluant une adresse de livraison.

```
<?xml version="1.0"?>
<purchaseOrder>
  <shipTo>
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
</purchaseOrder>
```

Voici le data model XForms concernant le précédant document XML - ordre d'achat:

```
<group name="purchaseOrder">
  <group name="shipTo">
    <string name="name"/>
    <string name="street"/>
    <string name="city"/>
    <string name="state"/>
    <string name="zip">
      <mask>ddddd</mask>
    </string>
  </group>
</group>
```

In XForms, permet de fusionner le data model et l'instance dans un document [XHTML 1.0] :

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML-XForms 1.0//EN"
"http://www.w3.org/TR/xhtml-forms1/DTD/xhtml-xforms1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Purchase Order</title>

<xform xmlns="http://www.w3.org/2000/xforms"
  action="http://www.my.com/cgi-bin/receiver.pl"
  method="postXML"
  id="po_xform">
  <model>
```

```
    <group name="purchaseOrder">
      <group name="shipTo">
        <string name="name"/>
        <string name="street"/>
        <string name="city"/>
        <string name="state"/>
        <string name="zip">
          <mask>ddddd</mask>
        </string>
      </group>
    </group>
  </model>
  <instance>
    <purchaseOrder>
      <shipTo>
        <name>Alice Smith</name>
        <street>123 Maple Street</street>
        <city>Mill Valley</city>
        <state>CA</state>
        <zip>90952</zip>
      </shipTo>
    </purchaseOrder>
  </instance>
</xform>
</head>

<body>
  <h1>Shipping Information</h1>

  <form name="po_xform">
    Name: <input name="purchaseOrder.shipTo.name"/><br/>
    Street: <input name="purchaseOrder.shipTo.street"/><br/>
    City: <input name="purchaseOrder.shipTo.city"/><br/>
    State: <input name="purchaseOrder.shipTo.state"/><br/>
    Zip: <input name="purchaseOrder.shipTo.zip"/><br/>
    <button onclick="submit('po_xform')">Submit</button>
  </form>
</body>
</html>
```

Note: ici les éléments standards de formulaire <input> <button> ont été utilises pour décrire l'interface utilisateur

- Quand l'utilisateur change le contenu des champs du formulaire, l'instance est mise a jour. Quand il soumet le formulaire les datas sont envoyés au serveur. Il est ensuite valide par SON Schema correspondant.
- XForms éditent le document XML dans le browser.

Exercice

1. Créer un document XML bien formé de type Répertoire:
   - composé de fiches (identifiant) incluant chacune:
   - nom, prénom, téléphone (localisation), email
2. Créer une DTD pour valider le document
3. Créer un Schema pour valider le document
4. Créer un feuille de style CSS permettant de l'afficher ce XML
5. Créer un feuille de style permettant de l'afficher en HTML (avec XSLT)
6. Créer un feuille de style permettant de l'afficher en XSL-FO (avec XSLT)
7. Créer un feuille de style permettant de créer un Index (avec XSLT)
8. Créer un feuille de style transformant le fichier XML repertoire (les éléments en attributs)

Fichier XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="repertoire2html.xsl"?>
<!DOCTYPE repertoire SYSTEM "repertoire.dtd">

<repertoire>
        <fiche numero="a020">
                <nom>Dubois</nom>
                <prenom>Paul</prenom>
                <telephone loc="home">04 56 43 67 54</telephone>
                <email>pd@provider.com</email>
        </fiche>

        <fiche numero="a021">
                <nom>Dupond</nom>
                <prenom>jean</prenom>
                <telephone loc="bureau">04 19 62 53 72</telephone>
                <email>jd@fournisseur.fr</email>
        </fiche>

        <fiche numero="a022">
                <nom>Deschamp</nom>
                <prenom>Isabelle</prenom>
                <telephone loc="portable">06 32 43 98 68</telephone>
                <email>id@acces.com</email>
        </fiche>
        <fiche numero="a023">
                <nom>Dubois</nom>
                <prenom>Alain</prenom>
                <telephone loc="portable">06 24 33 24 38</telephone>
                <email>ad@acces.com</email>
        </fiche>
</repertoire>
```

Déclaration de type de documents (repertoire.dtd)

```
<!ELEMENT repertoire (fiche)*>
<!ELEMENT fiche (nom,prenom,telephone,email)>
```

```
<!ATTLIST fiche numero ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT telephone (#PCDATA)>
<!ATTLIST telephone loc (home | bureau | portable) #REQUIRED>
<!ELEMENT email (#PCDATA)>
```

Fichier XML avec DTD incluse

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="repertoire2html.xsl"?>
<!DOCTYPE repertoire [
<!ELEMENT repertoire (fiche)*>
<!ELEMENT fiche (nom,prenom,telephone,email)>
<!ATTLIST fiche numero ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT telephone (#PCDATA)>
<!ATTLIST telephone loc (home | bureau | portable) #REQUIRED>
<!ELEMENT email (#PCDATA)>
]>

<repertoire>
        <fiche numero="a020">
                <nom>Dubois</nom>
                <prenom>Paul</prenom>
                <telephone loc="home">04 56 43 67 54</telephone>
                <email>pd@provider.com</email>
        </fiche>

        <fiche numero="a021">
                <nom>Dupond</nom>
                <prenom>jean</prenom>
                <telephone loc="bureau">04 19 62 53 72</telephone>
                <email>jd@fournisseur.fr</email>
        </fiche>

        <fiche numero="a022">
                <nom>Deschamp</nom>
                <prenom>Isabelle</prenom>
                <telephone loc="portable">06 32 43 98 68</telephone>
                <email>id@acces.com</email>
        </fiche>
        <fiche numero="a023">
                <nom>Dubois</nom>
                <prenom>Alain</prenom>
                <telephone loc="portable">06 24 33 24 38</telephone>
                <email>ad@acces.com</email>
        </fiche>
</repertoire>
```

Schema

Fichier XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="repertoire2html.xsl"?>
```

```
    <repertoire xmlns="http://www.exemple.org/XMLSchemarepertoire">
       ....
    </repertoire>


<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">

 <xsd:annotation>
  <xsd:documentation>
   schema de notre exemple Répertoire
  </xsd:documentation>
 </xsd:annotation>

 <xsd:element name="Repertoire"/>
 <!-- contenu de l'element Repertoire -->
 <xsd:complexType name="Repertoire" content='elementOnly'>

  <xsd:element name="Fiche" minOccurs="0" maxOccurs="*">
   <!-- contenu de l'element fiche -->
   <xsd:complexType content='elementOnly'>
    <xsd:element name="nom"    type="xsd:string"/>
    <xsd:element name="prenom" type="xsd:string"/>
    <xsd:element name="telephone">
       <!-- contenu de l'element telephone -->
       <xsd:complexType>
        <xsd:attribute name="loc">
          <!-- contenu de l'attribut loc -->
          <xsd:simpleType base="xsd:string">
            <xsd:enumeration  value="home"/>
            <xsd:enumeration  value="bureau"/>
            <xsd:enumeration  value="portable"/>
          </xsd:simpleType>
        </xsd:attribute>
       </xsd:complexType>
    </xsd:element
    <xsd:element name="email"  type="xsd:string"/>
    <xsd:attribute name="numero" type="xsd:ID" />
   </xsd:complexType>
  </xsd:element>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Schema : Syntaxe simplifiée

```
<?xml version='1.0'?>
<schema xmlns="http://www.w3.org/1999/XMLSchema"
        xmlns:rep="http://www.exemple.org/XMLSchemarepertoire"
        targetNamespace="http://www.exemple.org/XMLSchemarepertoire">


 <element name="Repertoire"/>
 <!-- contenu de l'element Repertoire -->
 <xsd:complexType name="Repertoire" content='elementOnly'>

  <element name="Fiche" minOccurs="0" maxOccurs="*">
   <!-- contenu de l'element fiche -->
   <complexType content='elementOnly'>
    <element name="nom"    type="string"/>
```

```
    <element name="prenom" type="string"/>
    <element name="telephone">
       <!-- contenu de l'element telephone -->
       <complexType>
       <xsd:attribute name="loc" type="rep:localisation"/>
       </complexType>
    </element
    <element name="email"  type="string"/>
    <attribute name="numero" type="ID" />
   </complexType>
  </element>
  </complexType>
 </element>


 <!-- contenu de l'attribut loc -->
 <simpleType name="rep:localisation" base="string">
   <enumeration  value="home"/>
   <enumeration  value="bureau"/>
   <enumeration  value="portable"/>
 </simpleType>

</schema>
```

Stylesheet CSS (repertoire.css)

```
repertoire {display:block; margin-left:10%;}
fiche {display:block; background:yellow; margin-left:2em;}
nom {font-weight:bolder; text-transform:uppercase;}
prenom,telephone,email {color:blue;}
```

autre exemple (ne fonctionne pas dans IE5:

```
repertoire    {display:table;}
fiche         {display:table-row; background:yellow;}
nom,prenom,telephone,email      {display:table-cell;}
```

Fichier XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="repertoire.css"?>
<repertoire> .... </repertoire>
```

demo: afficher le fichier repertoire avec CSS

Stylesheet XSLT (repertoire2html.xsl)

```
 <?xml version='1.0' encoding="ISO-8859-1"?>
   <xsl:stylesheet
         xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
         xmlns="http://www.w3.org/REC-html40"
         result="">
<xsl:template match="/">
```

```
<html lang="en">
<head>
<title>Repertoire de fiches</title>
</head>
    <body>
<h1>Répertoire téléphonique </h1>
        <table border="1">
            <tr>
                <th>Fiche</th>
                <th>Nom</th>
                <th>Prénom</th>
                <th>Téléphone</th>
                <th>email</th>
            </tr>
            <xsl:for-each select="repertoire/fiche">
                <!-- Trier par nom -->
                <xsl:sort select="nom"
                          data-type="text"
                          order="descending"/>
                <!-- Trier ensuite par prenom -->
                <xsl:sort select="prenom"/>
                <tr>
                    <td>
                        <em><xsl:value-of select="@numero"/></em>
                    </td>
                    <td>
                        <xsl:value-of select="nom"/>
                    </td>
                    <td>

                        <xsl:value-of select="prenom"/>
                    </td>
                    <td>
                    <!-- marquer les téléphones personnels en rouge -->
                           <xsl:if test="loc='home'">
                         <xsl:attribute name="style">
                           <xsl:text>color:red</xsl:text>
                         </xsl:attribute>
                       </xsl:if>
                        <xsl:value-of select="telephone"/>
                    </td>
                    <td>
                        <xsl:value-of select="email"/>
                    </td>
                </tr>
            </xsl:for-each>
        </table>
    </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Le fichier HTML résultant:

```
<html lang="en">
<head>
<title>Repertoire de fiches</title>
```

```
</head>
<body>
<h1>Répertoire téléphonique </h1>
<table border="1">

<tr>
<th>fiche</th><th>Nom</th><th>Prénom</th><th>Téléphone</th><th>email</th>
</tr>

<tr>
<td><em>022</em></td><td>Deschamp</td><td>Isabelle</td><td>06 32 43 98
68</td><td>id@acces.com</td>
</tr>
<tr>
<td><em>023</em></td><td>Dubois</td><td>Alain</td><td>06 24 33 24
38</td><td>ad@acces.com</td>
</tr>
<tr>
<td><em>020</em></td><td>Dubois</td><td>Paul</td><td>04 56 43 67
54</td><td>pd@provider.com</td>
</tr>

<tr>
<td><em>021</em></td><td>Dupond</td><td style="color:red">jean</td><td>04 19 62 53
72</td><td>jd@fournisseur.fr</td>
</tr>

</table>
</body>
</html>
```

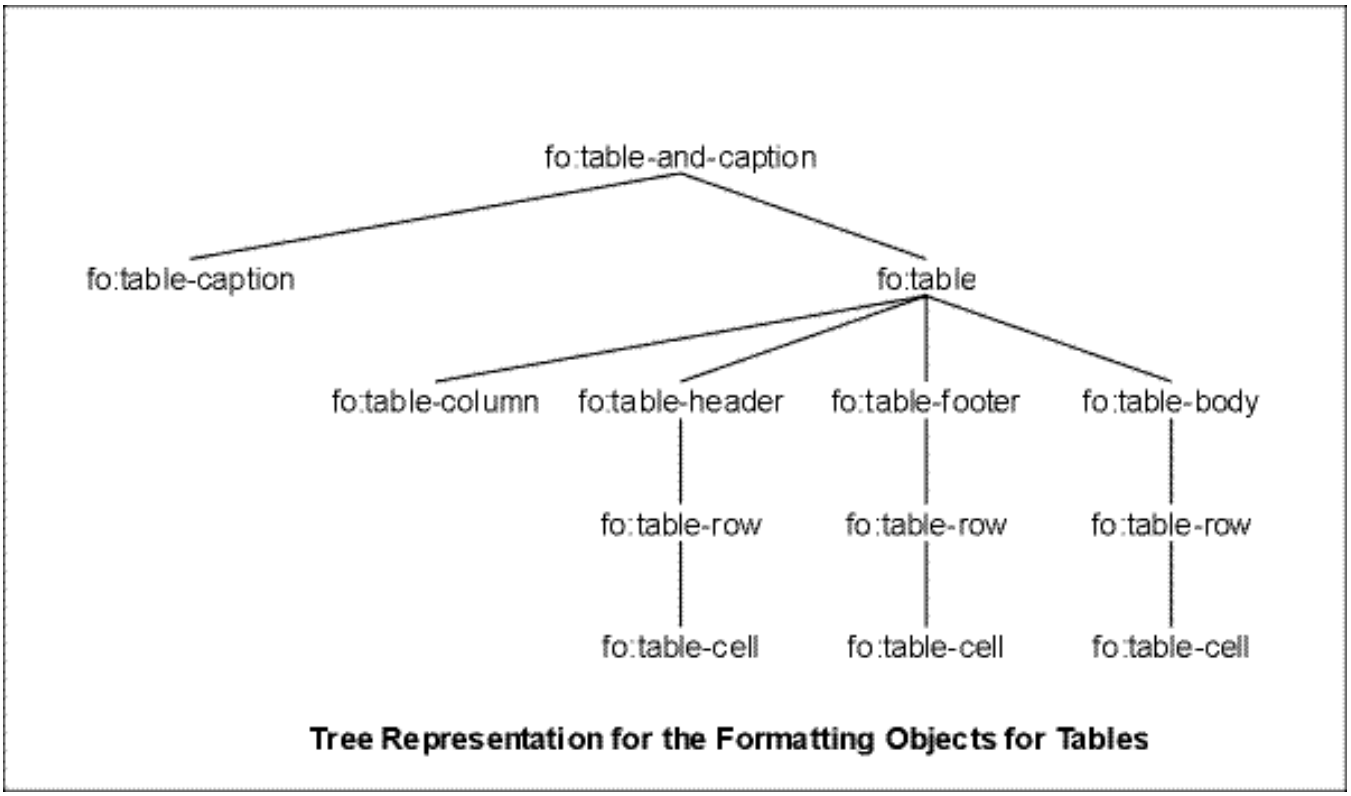
Le fichier HTML affiché

Répertoire téléphonique

fiche	Nom	Prénom	Téléphone	email
a022	Deschamp	Isabelle	06 32 43 98 68	id@acces.com
a023	Dubois	Alain	06 24 33 24 38	ad@acces.com
a020	Dubois	Paul	04 56 43 67 54	pd@provider.com
a021	Dupond	jean	04 19 62 53 72	jd@fournisseur.fr

demo: afficher le fichier repertoire avec XSL en HTML

Stylesheet XSLT (repertoire2fo.xsl)



Tree Representation for the Formatting Objects for Tables

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet
         xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
         xmlns:fo="http://www.w3.org/1999/XSL/FO"
         result="fo">
 <xsl:template match="/">
  <fo:basic-page-sequence font-family="arial" font-size="12pt">
 <fo:block > Répertoire téléphonique font-size="16pt"</fo:block>
<fo:table font-family="arial" font-size="12pt">
   <fo:table-header>
    <fo:table-row font-weight="bold" font-size="14pt">
       <fo:table-cell>Fiche</fo:table-cell>
       <fo:table-cell>Nom</fo:table-cell>
       <fo:table-cell>Prenom</fo:table-cell>
       <fo:table-cell>Telephone</fo:table-cell>
       <fo:table-cell>Email</fo:table-cell>
    </fo:table-row>
   </fo:table-header>
<fo:table-body>
     <xsl:for-each select="repertoire/fiche">

        <!-- Trier par nom -->
        <xsl:sort select="nom" data-type="text" order="descending"/>
        <!-- Trier ensuite par prenom -->
        <xsl:sort select="prenom"/>
     <fo:table-row>
      <fo:table-cell font-style="italic">
       <xsl:value-of select="@numero"/>
       </fo:table-cell>
       <fo:table-cell>
       <xsl:value-of select="nom"/>
       </fo:table-cell>
```

```
      <fo:table-cell>
      <xsl:value-of select="prenom"/>
      </fo:table-cell>

      <!-- marquer les téléphones personnels en rouge -->
      <xsl:if test="loc='home'">
         <xsl:attribute name="style">
           <xsl:text>color:red</xsl:text>
         </xsl:attribute>
      </xsl:if>
      <fo:table-cell>
      <xsl:value-of select="telephone"/>
      </fo:table-cell>
      <fo:table-cell>
      <xsl:value-of select="email"/>
      </fo:table-cell>

    </fo:table-row>
   </xsl:for-each>
  </fo:table-body>
 </fo:table>
</fo:basic-page-sequence>
</xsl:template>
</xsl:stylesheet>
```

Le fichier XSL-FO résultant:

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

<fo:basic-page-sequence font-family="arial" font-size="12pt">
<fo:block > Répertoire téléphonique font-size="16pt"</fo:block>
<fo:table font-family="arial" font-size="12pt">

  <fo:table-header>
    <fo:table-row font-weight="bold" font-size="14pt">
       <fo:table-cell>Fiche</fo:table-cell>
       <fo:table-cell>Nom</fo:table-cell>
       <fo:table-cell>Prenom</fo:table-cell>
       <fo:table-cell>Telephone</fo:table-cell>
       <fo:table-cell>Email</fo:table-cell>
    </fo:table-row>
  </fo:table-header>
<fo:table-body>
    <fo:table-row>
      <fo:table-cell font-style="italic">a022</fo:table-cell>
       <fo:table-cell>Deschamp</fo:table-cell>
       <fo:table-cell>Isabelle</fo:table-cell>
       <fo:table-cell>06 32 43 98 68</fo:table-cell>
       <fo:table-cell>id@acces.com</fo:table-cell>
    </fo:table-row>
  <fo:table-row>
      <fo:table-cell font-style="italic">a023</fo:table-cell>
       <fo:table-cell>Dubois</fo:table-cell>
       <fo:table-cell>Alain</fo:table-cell>
       <fo:table-cell>06 24 33 24 3</fo:table-cell>
       <fo:table-cell>ad@acces.com</fo:table-cell>
    </fo:table-row>
```

```
    <fo:table-row>
      <fo:table-cell font-style="italic">a020</fo:table-cell>
       <fo:table-cell>Dubois</fo:table-cell>
       <fo:table-cell>Paul</fo:table-cell>
       <fo:table-cell>04 56 43 67 54</fo:table-cell>
       <fo:table-cell>pd@provider.com</fo:table-cell>
    </fo:table-row>
    <fo:table-row>
      <fo:table-cell font-style="italic">a021</fo:table-cell>
       <fo:table-cell>Dupond</fo:table-cell>
       <fo:table-cell>Jean</fo:table-cell>
       <fo:table-cell>04 19 62 53 72</fo:table-cell>
       <fo:table-cell>jd@fournisseur.fr</fo:table-cell>
    </fo:table-row>
   </fo:table-body>
  </fo:table>
 </fo:basic-page-sequence>
```

Le fichier XSL-FO affiché

Répertoire téléphonique

fiche	Nom	Prénom	Téléphone	email
a022	Deschamp	Isabelle	06 32 43 98 68	id@acces.com
a023	Dubois	Alain	06 24 33 24 38	ad@acces.com
a020	Dubois	Paul	04 56 43 67 54	pd@provider.com
a021	Dupond	jean	04 19 62 53 72	jd@fournisseur.fr

Stylesheet XSLT (repertoire22html.xsl) avec xsl:apply-templates

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org/REC-html40" >

<xsl:template match="/">

<html lang="en">
 <head>
   <title>Repertoire de fiches</title>
 </head>
 <body style="background: white">
  <h1>Répertoire téléphonique </h1>

   <table border="1">
     <tr>
     <th>Fiche</th>
    <th>Nom</th>
```

```
    <th>Prénom</th>
   <th>Téléphone</th>
   <th>email</th>
  </tr>

  <xsl:apply-templates/>
 </table> </body>
</html>
</xsl:template>

<xsl:template match="fiche">
<tr>
<td><xsl:value-of select="@numero"/></td>
<xsl:apply-templates />
</tr>
</xsl:template>

<xsl:template match="nom">
<td><xsl:value-of select="."/></td>
</xsl:template>

<xsl:template match="prenom">
<td><xsl:value-of select="."/></td>
</xsl:template>

<xsl:template match="telephone">
<td><xsl:value-of select="."/></td>
</xsl:template>

<xsl:template match="email">
<td><xsl:value-of select="."/></td>
</xsl:template>

</xsl:stylesheet>
```

`</xsl:stylesheet>`

demo: afficher le fichier repertoire avec XSL en HTML

Transformation de repertoire XML en un autre fichier XML (sans éléments)

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet
         xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
         result="">

<xsl:template match="/">
<xsl:apply-templates select="repertoire"/>
</xsl:template>
  <xsl:template match="fiche">
   <xsl:copy>
    <xsl:apply-templates select="@numero"/>

    <xsl:attribute name="nom">
     <xsl:value-of select="nom/text"/>
    </xsl:attribute>
     <xsl:attribute name="prenom">
```

```
      <xsl:value-of select="prenom/text"/>
    </xsl:attribute>

    <xsl:attribute name="email">
      <xsl:value-of select="email/text"/>
    </xsl:attribute>

    <xsl:apply-templates select="telephone"/>
 </xsl:copy>
</xsl:template>

</xsl:stylesheet>
```

Fichier XML résultant:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="repertoire2html.xsl"?>
<!DOCTYPE repertoire SYSTEM "repertoire.dtd">

<repertoire>
  <fiche numero="a020" nom="Dubois"   prenom="Paul"     telephone="04 56 43 67 54"
loc="home"     email="pd@provider.com"></fiche>
  <fiche numero="a021" nom="Dupond"   prenom="jean"     telephone="04 19 62 53 72"
loc="bureau"   email="jd@fournisseur.fr"></fiche>
  <fiche numero="a022" nom="Deschamp" prenom="Isabelle" telephone="06 32 43 98 68"
loc="portable" email="pd@provider.com"></fiche>
  <fiche numero="a023" nom="Dubois"   prenom="Alain"    telephone="06 24 33 24 38"
loc="portable" email="ad@acces.com"></fiche>
</repertoire>
```

Transformation d'un fichier XML en deux représentations: HTML et SVG .

1- Le fichier XML

```
<sales>

        <division id="North">
                <revenue>10</revenue>
                <growth>9</growth>
                <bonus>7</bonus>
        </division>

        <division id="South">
                <revenue>4</revenue>
                <growth>3</growth>
                <bonus>4</bonus>
        </division>

        <division id="West">
                <revenue>6</revenue>
                <growth>-1.5</growth>
                <bonus>2</bonus>
        </division>

</sales>
```

2. Le style sheet de transformation en HTML:

```
<html xsl:version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      lang="en">
    <head>
        <title>Sales Results By Division</title>
    </head>
    <body>
        <table border="1">
            <tr>
                <th>Division</th>
                <th>Revenue</th>
                <th>Growth</th>
                <th>Bonus</th>
            </tr>
            <xsl:for-each select="sales/division">

                <tr>
                    <td>
                        <em><xsl:value-of select="@id"/></em>
                    </td>
```

```
                    <td>
                        <xsl:value-of select="revenue"/>
                    </td>
                    <td>

                        <xsl:value-of select="growth"/>
                    </td>
                    <td>
                        <xsl:value-of select="bonus"/>
                    </td>
                </tr>
            </xsl:for-each>
        </table>
    </body>
</html>
```

Le fichier HTML résultant:

```
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Sales Results By Division</title>
</head>
<body>
<table border="1">
<tr>
<th>Division</th><th>Revenue</th><th>Growth</th><th>Bonus</th>
</tr>
<tr>
<td><em>North</em></td><td>10</td><td>9</td><td>7</td>
</tr>
<tr>
<td><em>West</em></td><td>6</td><td>-1.5</td><td>2</td>
</tr>
<tr>
<td><em>South</em></td><td>4</td><td>3</td><td>4</td>
</tr>
</table>
</body>
</html>
```

2. Le style sheet de transformation en   SVG:

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://www.w3.org/Graphics/SVG/SVG-19990812.dtd">

<xsl:output method="xml" indent="yes" media-type="image/svg"/>

<xsl:template match="/">
```

```
<svg width = "3in" height="3in">
    <g style = "stroke: #000000">
        <!-- draw the axes -->
        <line x1="0" x2="150" y1="150" y2="150"/>
        <line x1="0" x2="0" y1="0" y2="150"/>
        <text x="0" y="10">Revenue</text>
        <text x="150" y="165">Division</text>
        <xsl:for-each select="sales/division">
            <!-- define some useful variables -->

            <!-- the bar's x position -->
            <xsl:variable name="pos"
                          select="(position()*40)-30"/>

            <!-- the bar's height -->
            <xsl:variable name="height"
                          select="revenue*10"/>

            <!-- the rectangle -->
            <rect x="{$pos}" y="{150-$height}"
                  width="20" height="{$height}"/>

            <!-- the text label -->
            <text x="{$pos}" y="165">
                <xsl:value-of select="@id"/>
            </text>

            <!-- the bar value -->
            <text x="{$pos}" y="{145-$height}">
                <xsl:value-of select="revenue"/>
            </text>
        </xsl:for-each>
    </g>
</svg>

</xsl:template>
</xsl:stylesheet>
```

Le fichier SVG résultant :

```
<svg width="3in" height="3in"
     xmlns="http://www.w3.org/Graphics/SVG/svg-19990412.dtd">
    <g style="stroke: #000000">
        <line x1="0" x2="150" y1="150" y2="150"/>
        <line x1="0" x2="0" y1="0" y2="150"/>
        <text x="0" y="10">Revenue</text>
        <text x="150" y="165">Division</text>
        <rect x="10" y="50" width="20" height="100"/>
        <text x="10" y="165">North</text>
        <text x="10" y="45">10</text>
```

```
        <rect x="50" y="110" width="20" height="40"/>
        <text x="50" y="165">South</text>
        <text x="50" y="105">4</text>
        <rect x="90" y="90" width="20" height="60"/>
        <text x="90" y="165">West</text>
        <text x="90" y="85">6</text>
    </g>
</svg>
```

Demo: fichier XML/XSL -->HTML

Demo: fichier SVG

Exemple Schema:

1. Le document XML (Fichier de Commande)

```
<?xml version="1.0"?>
<purchaseOrder
  xmlns="http://www.example.com/Order"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xsi:schemaLocation="http://www.example.com/Order
  http://www.example.com/Order.xsd"
  orderDate="1999-10-20">

    <shipTo country="US">
        <name>Alice Smith</name>
        <street>123 Maple Street</street>
        <city>Mill Valley</city>
        <state>CA</state>
        <zip>90952</zip>
    </shipTo>

    <billTo country="US">
        <name>Robert Smith</name>
        <street>8 Oak Avenue</street>
        <city>Old Town</city>
        <state>PA</state>
        <zip>95819</zip>
    </billTo>

    <comment>Hurry, my lawn is going wild!</comment>

    <items>
        <item partNum="872-AA">
            <productName>Lawnmower</productName>
            <quantity>1</quantity>
            <price>148.95</price>
            <comment>Confirm this is electric</comment>
        </item>
        <item partNum="926-AA">
            <productName>Baby Monitor</productName>
            <quantity>1</quantity>
            <price>39.98</price>
            <shipDate>1999-05-21</shipDate>
        </item>
    </items>

</purchaseOrder>
```

2. Le Schema XML (http://www.example.com/Order.xsd)

```
<xsd:schema targetNamespace="http://www.example.com/Order"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema">

<-- Annotations: lecteurs et applications -->
<xsd:annotation>
   <xsd:documentation>
     schema de bon de commande pour Example.com.
     Copyright 2000 Example.com. All rights reserved.
   </xsd:documentation>
</xsd:annotation>

<-- Element  Declarations -->

 <xsd:element name="purchaseOrder" type="PurchaseOrderType"/>

 <xsd:element name="comment" type="xsd:string"/>

<--    Complex Type Definitions  -->

 <xsd:complexType name="PurchaseOrderType">
  <xsd:element name="shipTo" type="Address"/>
  <xsd:element name="billTo" type="Address"/>
  <xsd:element ref="comment" minOccurs="0"/>
  <xsd:element name="items"  type="Items"/>
  <xsd:attribute name="orderDate" type="xsd:date"/>
 </xsd:complexType>

 <xsd:complexType name="Address">
  <xsd:element name="name"   type="xsd:string"/>
  <xsd:element name="street" type="xsd:string"/>
  <xsd:element name="city"   type="xsd:string"/>
  <xsd:element name="state"  type="xsd:string"/>
  <xsd:element name="zip"    type="xsd:decimal"/>
  <xsd:attribute name="country" type="xsd:NMTOKEN" fixed="US"/>
 </xsd:complexType>

<--   Anonymous Type Definitions -->

 <xsd:complexType name="Items">
  <xsd:element name="item" minOccurs="0" maxOccurs="*">
   <xsd:complexType>
    <xsd:element name="productName" type="xsd:string"/>
    <xsd:element name="quantity">
     <xsd:simpleType base="xsd:positive-integer">
      <xsd:maxExclusive value="100"/>
     </xsd:simpleType>
    </xsd:element>
```

```
    <xsd:element name="price"    type="xsd:decimal"/>
    <xsd:element ref="comment"   minOccurs="0"/>
    <xsd:element name="shipDate" type="xsd:date" minOccurs='0'/>
   <xsd:attribute name="partNum" type="Sku"/>
   </xsd:complexType>
  </xsd:element>
 </xsd:complexType>

<----   Simple Type Definitions  ---->

<xsd:simpleType name="Sku" base="xsd:string">
  <xsd:pattern value="/d{3}-[A-Z]{2}"/>
 </xsd:simpleType>

</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XML Spy v3.0.7 (http://www.xmlspy.com)-->
<!DOCTYPE xsd:schema PUBLIC "-//W3C//DTD XMLSCHEMA 19991216//EN" "" [
        <!ENTITY % p 'xsd:'>
        <!ENTITY % s ':xsd'>
]>
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
        <xsd:element name="dossier">
                <xsd:complexType content="elementOnly">
                        <xsd:sequence minOccurs="1" maxOccurs="unbounded">
                                <xsd:element name="fichier" type="fichierType"/>
                        </xsd:sequence>
                        <xsd:attribute name="xmlns:xsi" type="xsd:uriReference"
use="default" value="http://www.w3.org/1999/XMLSchema-instance"/>
                        <xsd:attribute name="xsi:noNamespaceSchemaLocation"
type="xsd:string"/>
                        <xsd:attribute name="xsi:schemaLocation" type="xsd:string"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:complexType name="fichierType" content="elementOnly">
                <xsd:sequence>
                        <xsd:element name="nom" type="xsd:string"/>
                        <xsd:element name="objet" type="xsd:string"/>
                </xsd:sequence>
                <xsd:attribute name="numero" type="xsd:ID" use="required"/>
        </xsd:complexType>
</xsd:schema>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<dossier xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="dossier.xsd">
        <fichier numero="n100">
                <nom>HTML</nom>
                <objet>HyperTexte Markup language</objet>
        </fichier>
        <fichier numero="n101">
                <nom>XML</nom>
                <objet>eXtensible Markup language</objet>
        </fichier>
        <fichier numero="n102">
                <nom>XSL</nom>
                <objet>eXtensible Stylesheet language</objet>
        </fichier>
</dossier>
```

XML Schema Include

The International Purchase Order Schema, ipo.xsd

```
<schema targetNamespace="http://www.example.com/IPO"
xmlns="http://www.w3.org/2000/08/XMLSchema"
xmlns:ipo="http://www.example.com/IPO">

<annotation>
<documentation>
International Purchase order schema for Example.com
Copyright 2000 Example.com. All rights reserved.
</documentation>
</annotation>

<!-- include address constructs -->
<include
schemaLocation="http://www.example.com/schemas/address.xsd"/>

<element name="purchaseOrder" type="ipo:PurchaseOrderType"/>

<element name="comment" type="string"/>

<complexType name="PurchaseOrderType">
<sequence>
<element name="shipTo" type="ipo:Address"/>
<element name="billTo" type="ipo:Address"/>
<element ref="ipo:comment" minOccurs="0"/>
<element name="items" type="ipo:Items"/>
</sequence>
<attribute name="orderDate" type="date"/>
</complexType>

<complexType name="Items">
<sequence>
<element name="item" minOccurs="0" maxOccurs="unbounded">
<complexType>
<sequence>
<element name="productName" type="string"/>
<element name="quantity">
<simpleType>
<restriction base="positiveInteger">
<maxExclusive value="100"/>
</restriction>
</simpleType>
</element>
```

```
<element name="USPrice" type="decimal"/>
<element ref="ipo:comment" minOccurs="0"/>
<element name="shipDate" type="date" minOccurs="0"/>
</sequence>
<attribute name="partNum" type="ipo:SKU"/>
</complexType>
</element>
</sequence>
</complexType>

<simpleType name="SKU">
<restriction base="string">
<pattern value="\d{3}-[A-Z]{2}"/>
</restriction>
</simpleType>

</schema>
```

Le fichier contenant le typage de address : (http://www.example.com/schemas/address.xsd)

```
<schema targetNamespace="http://www.example.com/IPO"
xmlns="http://www.w3.org/2000/08/XMLSchema"
xmlns:ipo="http://www.example.com/IPO">

<annotation>
<documentation>
Addresses for International Purchase order schema
Copyright 2000 Example.com. All rights reserved.
</documentation>
</annotation>

<complexType name="Address">
<sequence>
<element name="name" type="string"/>
<element name="street" type="string"/>
<element name="city" type="string"/>
</sequence>
</complexType>

<complexType name="USAddress">
<complexContent>
<extension base="ipo:Address">
<sequence>
<element name="state" type="ipo:USState"/>
<element name="zip" type="positiveInteger"/>
</sequence>
```

```
</extension>
</complexContent>
</complexType>

<complexType name="UKAddress">
<complexContent>
<extension base="ipo:Address">
<sequence>
<element name="postcode" type="ipo:UKPostcode"/>
</sequence>
<attribute name="exportCode" type="positiveInteger"
use="fixed" value="1"/>
</extension>
</complexContent>
</complexType>

<!-- other Address derivations for more countries -->

<simpleType name="USState">
<restriction base="string">
<enumeration value="AK"/>
<enumeration value="AL"/>
<enumeration value="AR"/>
<!-- and so on ... -->
</restriction>
</simpleType>

<!-- simple type definition for UKPostcode -->

</schema>
```

The purchase order schema is contained in the file po.xsd:

The Purchase Order Schema, po.xsd

```
<xsd:schema xmlns:xsd="http://www.w3.org/2000/08/XMLSchema">

<xsd:annotation>

<xsd:documentation>

Purchase order schema for Example.com.

Copyright 2000 Example.com. All rights reserved.

</xsd:documentation>

</xsd:annotation>

<xsd:element name="purchaseOrder" type="PurchaseOrderType"/>

<xsd:element name="comment" type="xsd:string"/>
```

```
<xsd:complexType name="PurchaseOrderType">
<xsd:sequence>
<xsd:element name="shipTo" type="USAddress"/>
<xsd:element name="billTo" type="USAddress"/>
<xsd:element ref="comment" minOccurs="0"/>
<xsd:element name="items" type="Items"/>
</xsd:sequence>
<xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>
<xsd:complexType name="USAddress">
<xsd:sequence>
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="street" type="xsd:string"/>
<xsd:element name="city" type="xsd:string"/>
<xsd:element name="state" type="xsd:string"/>
<xsd:element name="zip" type="xsd:decimal"/>
</xsd:sequence>
<xsd:attribute name="country" type="xsd:NMTOKEN"
use="fixed" value="US"/>
</xsd:complexType>
<xsd:complexType name="Items">
<xsd:sequence>
<xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="productName" type="xsd:string"/>
<xsd:element name="quantity">
<xsd:simpleType>
<xsd:restriction base="xsd:positiveInteger">
<xsd:maxExclusive value="100"/>
```

```
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="USPrice" type="xsd:decimal"/>
<xsd:element ref="comment" minOccurs="0"/>
<xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
</xsd:sequence>
<xsd:attribute name="partNum" type="SKU"/>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<!-- Stock Keeping Unit, a code for identifying products -->
<xsd:simpleType name="SKU">
<xsd:restriction base="xsd:string">
<xsd:pattern value="\d{3}-[A-Z]{2}"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

XML Schema Import

The Report Schema, report.xsd

```
<schema targetNamespace="http://www.example.com/Report"
xmlns="http://www.w3.org/2000/08/XMLSchema"
xmlns:r="http://www.example.com/Report"
xmlns:xipo="http://www.example.com/IPO" >

<import namespace="http://www.example.com/IPO"
        schemaLocation="http://www.example.com/IPO/ipo.xsd"/>

<annotation>
<documentation>
Report schema for Example.com
Copyright 2000 Example.com. All rights reserved.
</documentation>
</annotation>

<element name="purchaseReport">
<complexType>
<sequence>
<element name="regions" type="r:RegionsType"/>
<element name="parts" type="r:PartsType"/>
</sequence>
<attribute name="period" type="timeDuration"/>
<attribute name="periodEnding" type="date"/>
</complexType>

<unique name="dummy1">
<selector xpath="regions/zip"/>
<field xpath="@code"/>
</unique>

<key name="pNumKey">
<selector xpath="parts/part"/>
<field xpath="@number"/>
</key>

<keyref name="dummy2" refer="pNumKey">
<selector xpath="regions/zip/part"/>
<field xpath="@number"/>
</keyref>
</element>

<complexType name="RegionsType">
<sequence>
```

```
<element name="zip" maxOccurs="unbounded">
<complexType>
<sequence>
<element name="part" maxOccurs="unbounded">
<complexType>
<complexContent>
<restriction base="anyType">
<attribute name="number" type="xipo:SKU"/>
<attribute name="quantity" type="positiveInteger"/>
</restriction>
</complexContent>
</complexType>
</element>
</sequence>
<attribute name="code" type="positiveInteger"/>
</complexType>
</element>
</sequence>
</complexType>

<complexType name="PartsType">
<sequence>
<element name="part" maxOccurs="unbounded">
<complexType>
<simpleContent>
<extension base="string">
<attribute name="number" type="xipo:SKU"/>
</extension>
</simpleContent>
</complexType>
</element>
</sequence>
</complexType>

</schema>
```

Exemples : Transformation de XML en XSL-FO avec XSLT

A2- Document source XML Réguliers (répétition d'éléments type, BdD)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="dossier2xsl-fo.xsl"?>

     <dossier>
         <fichier numero="n100">
           <nom>HTML</nom>
           <objet>HyperTexte Markup language</objet>
        </fichier>
         <fichier numero="n101">
           <nom>XML</nom>
           <objet>eXtensible Markup language</objet>
        </fichier>

          <fichier numero="n102">
           <nom>XSL</nom>
           <objet>eXtensible Stylesheet language</objet>
         </fichier>

     </dossier>
```

2. Stylesheet XSLT (dossier2xsl-fo.xsl)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
   <?xml version="1.0" encoding="ISO-8859-1"?>
   <xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:fo="http://www.w3.org/1999/XSL/Format">


<xsl:template match="/">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="page" page-height="297mm"
page-width="210mm" margin-top="20mm" margin-bottom="10mm" margin-left="20mm"
margin-right="20mm">
         <fo:region-body region-name="main" margin-top="0mm" margin-bottom="10mm"
margin-left="0mm" margin-right="0mm"/>
      </fo:simple-page-master>
   </fo:layout-master-set>
   <fo:page-sequence master-name="page">
      <fo:flow flow-name="main">
   <fo:list-block font-size="17pt">

   <xsl:for-each select="dossier/fichier">

    <fo:list-item>

        <fo:list-item-label font-weight="bold" font-size="12pt">
        <fo:block>Numéro:</fo:block>
       </fo:list-item-label>

       <fo:list-item-body start-indent="20mm" font-style="italic" font-size="12pt">
       <fo:block><xsl:value-of select="@numero"/></fo:block>
       </fo:list-item-body>
        </fo:list-item>
```

```
        <fo:list-item>

       <fo:list-item-label  font-weight="bold" font-size="12pt">
        <fo:block>Nom:</fo:block>
       </fo:list-item-label>

       <fo:list-item-body start-indent="15mm" color="red" font-style="italic"
font-size="12pt">
       <fo:block><xsl:value-of select="nom"/></fo:block>
       </fo:list-item-body>
        </fo:list-item>
        <fo:list-item>

       <fo:list-item-label font-weight="bold" font-size="12pt">
        <fo:block>Objet:</fo:block>
        </fo:list-item-label>

        <fo:list-item-body start-indent="15mm" font-style="italic" font-size="12pt">
        <fo:block><xsl:value-of select="objet"/></fo:block>
        </fo:list-item-body>
     </fo:list-item>

      </xsl:for-each>
    </fo:list-block>
     </fo:flow>
   </fo:page-sequence>
  </fo:root>
 </xsl:template>
</xsl:stylesheet>
```

3. Le fichier XSL-FO en résultant:

```
<?xml version="1.0" encoding="UTF-16" ?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
 <fo:simple-page-master master-name="page" page-height="297mm" page-width="210mm"
margin-top="20mm" margin-bottom="10mm" margin-left="20mm" margin-right="20mm">
  <fo:region-body region-name="main" margin-top="0mm" margin-bottom="10mm"
margin-left="0mm" margin-right="0mm" />
  </fo:simple-page-master>
  </fo:layout-master-set>
 <fo:page-sequence master-name="page">
 <fo:flow flow-name="main">
 <fo:list-block font-size="17pt">

<fo:list-item>
 <fo:list-item-label font-weight="bold" font-size="12pt">
  <fo:block>Numero:</fo:block>
  </fo:list-item-label>
 <fo:list-item-body start-indent="20mm" font-style="italic" font-size="12pt">
  <fo:block>n100</fo:block>
  </fo:list-item-body>
  </fo:list-item>

<fo:list-item>
 <fo:list-item-label font-weight="bold" font-size="12pt">
  <fo:block>Nom:</fo:block>
  </fo:list-item-label>
 <fo:list-item-body start-indent="15mm" color="red" font-style="italic"
font-size="12pt">
```

```
  <fo:block>HTML</fo:block>
  </fo:list-item-body>
  </fo:list-item>

 <fo:list-item>
 <fo:list-item-label font-weight="bold" font-size="12pt">
  <fo:block>Objet:</fo:block>
  </fo:list-item-label>
 <fo:list-item-body start-indent="15mm" font-style="italic" font-size="12pt">
  <fo:block>HyperTexte Markup language</fo:block>
  </fo:list-item-body>
  </fo:list-item>

<fo:list-item>
 <fo:list-item-label font-weight="bold" font-size="12pt">
  <fo:block>Numero:</fo:block>
  </fo:list-item-label>
 <fo:list-item-body start-indent="20mm" font-style="italic" font-size="12pt">
  <fo:block>n101</fo:block>
  </fo:list-item-body>
  </fo:list-item>

 <fo:list-item>
 <fo:list-item-label font-weight="bold" font-size="12pt">
  <fo:block>Nom:</fo:block>
  </fo:list-item-label>
 <fo:list-item-body start-indent="15mm" color="red" font-style="italic"
font-size="12pt">
  <fo:block>XML</fo:block>
  </fo:list-item-body>
  </fo:list-item>

 <fo:list-item>
 <fo:list-item-label font-weight="bold" font-size="12pt">
  <fo:block>Objet:</fo:block>
  </fo:list-item-label>
 <fo:list-item-body start-indent="15mm" font-style="italic" font-size="12pt">
  <fo:block>eXtensible Markup language</fo:block>
  </fo:list-item-body>
  </fo:list-item>

 <fo:list-item>
 <fo:list-item-label font-weight="bold" font-size="12pt">
  <fo:block>Numero:</fo:block>
  </fo:list-item-label>
 <fo:list-item-body start-indent="20mm" font-style="italic" font-size="12pt">
  <fo:block>n102</fo:block>
  </fo:list-item-body>
  </fo:list-item>

 <fo:list-item>
 <fo:list-item-label font-weight="bold" font-size="12pt">
  <fo:block>Nom:</fo:block>
  </fo:list-item-label>
 <fo:list-item-body start-indent="15mm" color="red" font-style="italic"
font-size="12pt">
  <fo:block>XSL</fo:block>
  </fo:list-item-body>
  </fo:list-item>
```

```
<fo:list-item>
 <fo:list-item-label font-weight="bold" font-size="12pt">
  <fo:block>Objet:</fo:block>
  </fo:list-item-label>
 <fo:list-item-body start-indent="15mm" font-style="italic" font-size="12pt">
  <fo:block>eXtensible Stylesheet language</fo:block>
  </fo:list-item-body>
  </fo:list-item>
  </fo:list-block>
  </fo:flow>
  </fo:page-sequence>
  </fo:root>
```

4. Le fichier affiché

Numéro : *n100*
Nom : *HTML*
Objet : *HyperTexte Markup Language*

Numéro : *n101*
Nom : *XML*
Objet : *eXtensible Markup Language*

Numéro : *n102*
Nom : *XSL*
Objet : *eXtensible Stylesheet Language*

Demo Fichier Dossier affiche en FO dans XSL Formatter (ne fonctionne pas dans IE5)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="dossier2XSL-FO.xsl"?>


<!DOCTYPE dossier SYSTEM "dossier.dtd">
     <dossier>
        <fichier numero="n100">
           <nom>HTML</nom>
           <objet>HyperTexte Markup language</objet>
        </fichier>

        <fichier numero="n101">
           <nom>XML</nom>
           <objet>eXtensible Markup language</objet>
        </fichier>

         <fichier numero="n102">
           <nom>XSL</nom>
           <objet>eXtensible Stylesheet language</objet>
         </fichier>

     </dossier>
```

Transformation de XML en HTML avec XSLT

1- Document source XML Réguliers (répétition d'éléments type, BdD):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="dossier2html.xsl"?>

     <dossier>
        <fichier numero="n100">
           <nom>HTML</nom>
           <objet>HyperTexte Markup language</objet>
        </fichier>
       <fichier numero="n101">
           <nom>XML</nom>
           <objet>eXtensible Markup language</objet>
        </fichier>

         <fichier numero="n102">
           <nom>XSL</nom>
           <objet>eXtensible Stylesheet language</objet>
         </fichier>

     </dossier>
```

2. Stylesheet XSLT (dossier2html.xsl)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
   <xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns="http://www.w3.org/REC-html40">
<xsl:template match="/">
<html lang="en">
<head>
<title>Dossiers</title>
<style>
.dossier {font-family:arial;}
.numero {font-style:italic; font-size:12pt;}
.entete {font-weight:bold; font-size:14pt;}
.info {font-style:italic; font-size:12pt;}
</style>
</head>
<body>
  <table border="1">
```

```
    <tbody class="dossier">
    <xsl:for-each select="dossier/fichier">
      <tr>
          <th class="entete">Numéro:
          </th>
   <td class="numero">
     <xsl:value-of select="@numero"/>
        </td>

          <th class="entete">Nom:
          </th>
          <td class="info">
          <xsl:value-of select="nom"/>
          </td>

          <th class="entete">Objet:
          </th>
          <td class="info">
          <xsl:value-of select="objet"/>
          </td>
        </tr>
    </xsl:for-each>
  </tbody>
 </table>
 </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

3. Le fichier HTML en résultant:

```
<html lang="en">
<head>
<title>Dossiers</title>
<style>
.dossier {font-family:arial;}
.entete {font-weight:bold; font-size:14pt;}
.info {font-style:italic; font-size:12pt;}
</style>
</head>
<body>
  <table>
    <tbody class="dossier">

        <tr>
          <th class="entete">Numéro:
          </th>
          <td class="numero">
```

```
      n100
    </td>
    <th class="entete">Nom:
    </th>
    <td class="info">
     HTML
    </td>

    <th class="entete">Objet:
    </th>
    <td class="info">
     HyperTexte Markup Language
    </td>
  </tr>
  <tr>
    <th class="entete">Numéro:
    </th>
    <td class="numero">
     n101
    </td>
    <th class="entete">Nom:
    </th>
    <td class="info">
     XML
    </td>

    <th class="entete">Objet:
    </th>
    <td class="info">
     eXtensible Markup Language
    </td>
  </tr>
  <tr>
    <th class="entete">Numéro:
    </th>
    <td class="numero">
     n102
    </td>
    <th class="entete">Nom:
    </th>
    <td class="info">
     XSL
    </td>

    <th class="entete">Objet:
    </th>
    <td class="info">
   eXtensible Stylesheet Language
```

```
          </td>
        </tr>
    </tbody>
 </table>
 </body>
</html>
```

4. Le fichier affiché

Numéro :	*n100*	**Nom** :	*HTML*	**Objet** :	HyperTexte Markup Language
Numéro :	*n101*	**Nom** :	*XML*	**Objet** :	eXtensible Markup Language
Numéro :	*n102*	**Nom** :	*XSL*	**Objet** :	eXtensible Stylesheet Language

Demo : Fichier dossier.xml (exécution dans IE5)

B1:Transformation de XML en HTML avec XSLT

Document source XML non Réguliers (arbre d'éléments en profondeur):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="chapitre2html.xsl"?>
<chapitre>
        <section>
           <titre>Les feuilles de style XSL</titre>
           <paragraphe>Exemple d'utilisation d'une feuille de style
           <exergue>XSL</exergue>
           </paragraphe>
        </section>
</chapitre>
```

2- Stylesheet XSLT (chapitre2html.xsl)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
   <xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns="http://www.w3.org/REC-html40">

<xsl:template match="/">
 <html lang="en">
 <head>
  <style>
   .section {font-family:arial;}
   .titre {font-size:14pt; font-weight:bold;color:red;}
   .paragraphe {font-size:12pt;}
   .exergue {font-style:italic;}
  </style>
 </head>
 <body>
 <xsl:apply-templates/>
 </body></html>
</xsl:template>

<xsl:template match="section">
  <DIV class="section">
  <xsl:apply-templates/>
  </DIV>
</xsl:template>
```

```
<xsl:template match="titre">
  <DIV class="titre">
  <xsl:apply-templates/>
  </DIV>
</xsl:template>

<xsl:template match="paragraphe">
  <DIV class="paragraphe">
  <xsl:apply-templates/>
  </DIV>
</xsl:template>

<xsl:template match="exergue">
  <SPAN class="exergue">
  <xsl:apply-templates/>
  </SPAN>
</xsl:template>
</xsl:stylesheet>
```

3. Le fichier HTML résultant:

```
 <html lang="en">
 <head>
  <style>
   .section {font-family:arial;}
   .titre {font-size:14pt; font-weight:bold;}
   .paragraphe {font-size:12pt;}
   .exergue {font-style:italic;}
  </style>
 </head>
 <body>

  <DIV class="section">
    <DIV class="titre">Les feuilles de style XSL
    </DIV>
    <DIV class="paragraphe">
       Exemple d'utilisation d'une feuille de style
      <SPAN class="exergue">XSL</SPAN>
    </DIV>
  </DIV>
</body></html>
```

4. Le fichier affiché

Les feuilles de style XSL

Exemple d'utilisation d'une feuille de style *XSL*

voir le resultat dans IE5.5 (XSL/W3C)

voir le resultat dans IE5.5 (XSL/Microsoft)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="chapitre2html.xsl"?>
<chapitre>
        <section>
                <titre>les feuilles de style XSL10</titre>
                <paragraphe>Exemple d'utilisation d'une feuille de style
           <exergue>XSL</exergue>
                </paragraphe>
        </section>
</chapitre>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="chapitre2MShtml.xsl"?>
<chapitre>
        <section>
           <titre>Les feuilles de style XSL</titre>
           <paragraphe>Exemple d'utilisation d'une feuille de style
           <exergue>XSL</exergue>
           </paragraphe>
        </section>
</chapitre>
```

Transformation XSL exécutée sur le serveur

Exemple sur un serveur IIS

```
   <%@ LANGUAGE=JSCRIPT %>
   <%
//on charge le document XML source
   var source = Server.CreateObject("Microsoft.XMLDOM");
   source.load(Server.MapPath("document.xml"));

//on charge la feuille de style XSL
var style = Server.CreateObject("Microsoft.XMLDOM");
source.load(Server.MapPath("monstyle.xsl"));

//on transforme le document source
var result = source.transformNode(style);
Response.Write(result);
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="dossier.css"?>
<!DOCTYPE dossier SYSTEM "dossier.dtd">

<dossier>
        <fichier numero="n100">
                <nom>HTML</nom>
                <objet>HyperTexte Markup language</objet>
        </fichier>
        <fichier numero="n101">
                <nom>XML</nom>
                <objet>eXtensible Markup language</objet>
        </fichier>
        <fichier numero="n102">
                <nom>XSL</nom>
                <objet>eXtensible Stylesheet language</objet>
        </fichier>
</dossier>
```

```
dossier {display:block; margin-left:10%;}
fichier {display:block; background:yellow; margin-left:2em;}
nom {display:block;font-weight:bolder; text-transform:uppercase;}
objet {display:block;color:blue; margin-left:2em;}
```

Exemples de Xpointer:

1. Renvoi sur un nœud:
   Syntaxe :
   `Type de renvoi(occurrence, type de noeud, attribut, valeur-attribut)`

Type de renvoi:	child	descendant	ancestor	psibling/fsibling	preceding/following

Type de noeud	#élément	#pi	#comment	#text	#cdata

   - Child
   - `#root().child(-1, chapitre).child(3,section)`
   - *renvoi a la troisième section du dernier chapitre du document*
   - `.child(1, #element, diffusion, "confidentiel")`
     renvoi au premier élément rencontré ayant un attribut diffusion dont la valeur est "confidentiel"
   - `.child(1, adresse, cp, *)`
     renvoi a premier élément adresse ayant un attribut cp dont la valeur est quelconque
   - Psibling:
     `#origin().psibling(1)`
     renvoi a l'élément frère précédent
   - Descendant/ancestor
     `id(num240).descendant(2,nombre)`
     renvoi a la seconde occurrence de l'élément "nombre" a un niveau qq de profondeur dans l'élément ayant pour id num240
   - `.descendant(1,prenom).ancestor(1 ,organisation)`
     renvoi au premier l'élément "organisation" qui contient la première occurrence de l'élément "prenom"
   - `.child(1,emp).ancestor(1 ,p)` *renvoi au noeud "p" dans l'exemple suivant:*
     `<p> paragraphe avec<emp>section en valeur</emp></p>`
2. Renvoi sur une valeur d'attribut: (attr) sélectionner la valeur d'un attribut
   `#root().child(1, adresse.attr(cp)`
   sélectionne la valeur de l'attribut cp du premier élément adresse rencontré
3. Renvoi sur une chaîne: (string)
   `#root().string(1,foo).string(16,"")`
   sélectionne la position précédant le 16eme caractère contenu de l'élément foo

```
#root().string(3, "Thierry",1,7)
```
*sélectionne les 7 caractères qui suivent la position précédant le premier caractère de la troisième occurrence de Thierry cad "*Thierry"

4. Renvoi sur un intervalle: (span)

```
#root().id(t36) .span (child(1),child(3))
```
sélectionne les trois premiers éléments fils de celui ayant un id =t36

```
#root().child(1, caption).span (string(1,"exemple"), string(1,"2",end)
```
sélectionne dans le premier élément "caption" toute la chaîne de valeur débutant par "exemple"et finissant par deux

Un exemple de lien simple Xlink:

L'élément XLink pour les liens simples est de type "simple":

```
<lien_simple xmlns:xlink="http://www.w3.org/1999/xlink"
             xlink:type="simple"
              xlink:href="students.xml"
             xlink:title="Liste d'étudiants"
              xlink:role="http://www.exemple.com/studentlist"
              xlink:show="new"
             xlink:actuate="onRequest">Liste actuelle des étudiants
</lien_simple>
```

Autre syntaxe plus légère:

type =simple show="new" et actuate="onRequest" peuvent être fixés dans la DTD

Dans la DTD

```
<!ELEMENT lien_simple ANY>
<!ATTLIST lien_simple
         xlink:type NMTOKEN #FIXED "simple"
          xlink:show NMTOKEN #FIXED "new"
          xlink:actuate NMTOKEN #FIXED "onRequest"
          xlink:href CDATA #REQUIRED
          xlink:role CDATA #IMPLIED
          xlink:title CDATA #IMPLIED >
```

Dans l'Instance

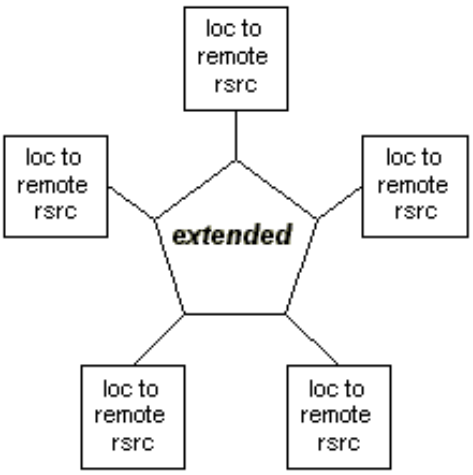
```
<lien_simple xmlns:xlink="http://www.w3.org/1999/xlink"
          xlink:href="students.xml"
         xlink:title="Liste d'étudiants"
          xlink:role="http://www.exemple.com/studentlist"
</lien_simple>
```

Dans un autre namespace:

Dans l'Instance

```
<my:lien_simple
  xmlns:my="http://example.com/"
  my:lastEdited="2000-11-25"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:type="simple"
  xlink:href="students.xml"
  xlink:role="http://www.exemple.com/studentlist"
  xlink:title="Liste d'étudiants"
  xlink:show="new"
  xlink:actuate="onRequest">
Liste actuelle des étudiants
</my:lien_simple>
```

Un exemple de lien multiple Xlink:



L'élément XLink pour les liens étendus est de type "extended".
Il contient les éléments suivants contenant les valeurs de l'attributs de type suivants:

- **locator** fournit l'adresse des ressources externes (destination)

```
<lien_multiple
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xlink:title="lien etendu"
   xlink:type="extented">
  <loc xlink:type="locator" xlink:href="http://www.foo.org/"
xlink:role="http://www.exemple.org/parent" xlink:title="p1" />
  <loc xlink:type="locator" xlink:href="http://www.ici.org/"
xlink:role="http://www.exemple.org/child"  xlink:title="c1" />
  <loc xlink:type="locator" xlink:href="http://www.labas.org/"
xlink:role="http://www.exemple.org/child"  xlink:title="c2" />
  <loc xlink:type="locator" xlink:href="http://www.ailleurs.org/"
xlink:role="http://www.exemple.org/child"  xlink:title="c3" />
</lien_multiple>
```

Autre syntaxe plus légère:

type = extented et type = locator peuvent être déclaré dans la DTD

```
<!-- extendedlink = extended-type -->
<!ELEMENT lien_multiple (loc*)>
<!ATTLIST lien_multiple
xlink:type NMTOKEN #FIXED "extended"
xlink:title CDATA #IMPLIED

<!ELEMENT loc EMPTY>
<!ATTLIST loc xlink:type NMTOKEN #FIXED "locator"
xlink:href CDATA #REQUIRED
xlink:role CDATA #IMPLIED
xlink:title CDATA #IMPLIED
-->
```

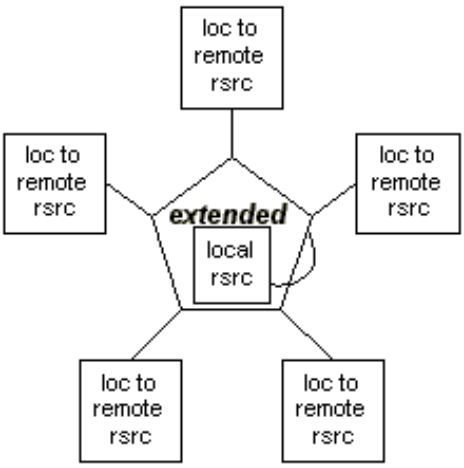
les éléments XML utilisant cette déclaration:

```
<lien_multiple  xmlns:xlink="http://www.w3.org/1999/xlink" xlink:title="lien etendu"

  <loc  xlink:href="http://www.foo.org/" xlink:title="p1" />
  <loc  xlink:href="http://www.ici.org/" xlink:title="c1" />
  <loc  xlink:href="http://www.labas.org/" xlink:title="c2" />
  <loc  xlink:href="http://www.ailleurs.org/" xlink:title="c3" />

</lien_multiple>
```

Exemple de lien multiple Xlink, utilisant les arcs



L'élément XLink pour les liens étendus peut contenir des éléments fils, contenant les valeurs de l'attributs de type suivants:

- **locator** fournit l'adresse des ressources externes (destination)
- **resource** fournit l'adresse des ressources locales (destination)
- **arc** fournit les ressouces participant au lien (a une paire d'attributs **from** et **to**)
- **title** fournit un titre lisible
- Ces éléments fils peuvent avoir des attributs sémantiques (**role**, **arcrole**, **label**, et **title**)).

Dans la DTD

```
<!ELEMENT courseload ((person|course|gpa|go)*)>
<!ATTLIST courseload
  xlink:type      NMTOKEN      #FIXED "extended"
  xlink:role      CDATA           #IMPLIED
  xlink:title     CDATA           #IMPLIED>

<!ELEMENT person EMPTY>
<!ATTLIST person
  xlink:type      NMTOKEN       #FIXED "locator"
  xlink:href      CDATA           #REQUIRED
  xlink:role      CDATA           #IMPLIED
  xlink:title     CDATA           #IMPLIED
  xlink:label     ID              #REQUIRED>

<!ELEMENT course EMPTY>
<!ATTLIST course
  xlink:type      NMTOKEN       #FIXED "locator"
  xlink:href      CDATA           #REQUIRED
  xlink:role      CDATA           #FIXED "http://www.example.com/linkprops/course"
  xlink:title     CDATA           #IMPLIED
  xlink:label     ID              #REQUIRED>

<!ELEMENT gpa ANY>
<!ATTLIST gpa
  xlink:type      NMTOKEN      #FIXED "resource"
  xlink:role      CDATA           #FIXED "http://www.example.com/linkprops/gpa"
  xlink:title     CDATA           #IMPLIED
  xlink:label     ID              #REQUIRED>

<!ELEMENT go EMPTY>
<!ATTLIST go
```

```
  xlink:type      NMTOKEN          #FIXED "arc"
  xlink:arcrole   CDATA            #IMPLIED
  xlink:title     CDATA            #IMPLIED
  xlink:show      (new|replace|embed) #IMPLIED

  xlink:actuate   NMTOKEN       #FIXED "onRequest"
  xlink:from      IDREF         #IMPLIED
  xlink:to        IDREF         #IMPLIED>
```

Dans l'instance XML:

```
<courseload xlink:title="Course Load for Pat Jones">

  <person
    xlink:href="students/patjones62.xml"
    xlink:label="student62"
    xlink:role="http://www.example.com/linkprops/student"
    xlink:title="Pat Jones" />

  <person
    xlink:href="profs/jaysmith7.xml"
    xlink:label="prof7"
    xlink:role="http://www.example.com/linkprops/professor"
    xlink:title="Dr. Jay Smith" />

  <!-- d'autres resources pour professor, etc. -->

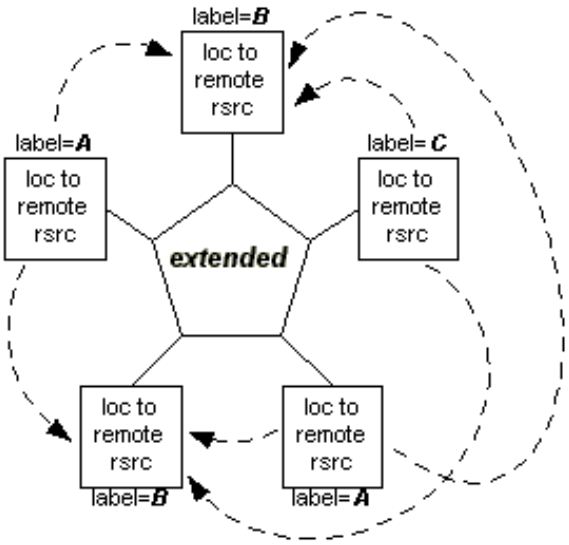
  <course
    xlink:href="courses/cs101.xml"
    xlink:label="CS-101"
    xlink:title="Computer Science 101" />
  <!-- more remote resources for courses, seminars, etc. -->

  <gpa xlink:label="PatJonesGPA">3.5</gpa>

  <go
    xlink:from="student62"
    xlink:to="PatJonesGPA"
    xlink:show="new"
    xlink:title="Pat Jones's GPA" />
  <go
    xlink:from="CS-101"
    xlink:arcrole="http://www.example.com/linkprops/auditor"
    xlink:to="student62"
    xlink:show="replace"
    xlink:title="Pat Jones, auditing the course" />
  <go
    xlink:from="student62"
    xlink:arcrole="http://www.example.com/linkprops/advisor"
    xlink:to="prof7"
    xlink:show="replace"
       xlink:title="Dr. Jay Smith, advisor" />

</courseload>
```

Autre exemple de lien multiple Xlink, utilisant les arcs



Les éléments de type **arc** définissent une paire de ressources par l'intermédiaire des attributs (**from, to**)

```
<go xlink:type="arc" xlink:from="A" xlink:to="B" />
```

Lien étendu utilisant les éléménts de type **locator** et **label**

```
<extendedlink xlink:type="extended">
  <loc xlink:type="locator" xlink:href="..." xlink:label="parent" xlink:title="p1" />
  <loc xlink:type="locator" xlink:href="..." xlink:label="parent" xlink:title="p2" />
  <loc xlink:type="locator" xlink:href="..." xlink:label="child"  xlink:title="c1" />
  <loc xlink:type="locator" xlink:href="..." xlink:label="child"  xlink:title="c2" />
  <loc xlink:type="locator" xlink:href="..." xlink:label="child"  xlink:title="c3" />

  <go xlink:type="arc" xlink:from="parent" xlink:to="child" />

</extendedlink>
```

Spécifie les liens des ressources "parent" vers "child" cad: p1-c1, p1-c2, p1-c3, p2-c1, p2-c2, and p2-c3:

```
<go xlink:type="arc" xlink:to="child" />
```

p1-c1, p1-c2, p1-c3, p2-c1, p2-c2, p2-c3, c1-c1, c1-c2, c1-c3, c2-c1, c2-c2, c2-c3, c3-c1, c3-c2, and c3-c3:

XPath: Exemples de location paths

Syntaxe non simplifie:

- `child::para` selects the `para` element children of the context node
- `child::*` selects all element children of the context node
- `child::text()` selects all text node children of the context node
- `child::node()` selects all the children of the context node, whatever their node type
- `attribute::name` selects the `name` attribute of the context node
- `attribute::*` selects all the attributes of the context node
- `descendant::para` selects the `para` element descendants of the context node
- `ancestor::div` selects all `div` ancestors of the context node
- `ancestor-or-self::div` selects the `div` ancestors of the context node and, if the context node is a `div` element, the context node as well
- `descendant-or-self::para` selects the `para` element descendants of the context node and, if the context node is a `para` element, the context node as well
- `self::para` selects the context node if it is a `para` element, and otherwise selects nothing
- `child::chapter/descendant::para` selects the `para` element descendants of the `chapter` element children of the context node
- `child::*/child::para` selects all `para` grandchildren of the context node
- `/` selects the document root (which is always the parent of the document element)
- `/descendant::para` selects all the `para` elements in the same document as the context node
- `/descendant::olist/child::item` selects all the `item` elements that have an `olist` parent and that are in the same document as the context node
- `child::para[position()=1]` selects the first `para` child of the context node
- `child::para[position()=last()]` selects the last `para` child of the context node
- `child::para[position()=last()-1]` selects the last but one `para` child of the context node
- `child::para[position()>1]` selects all the `para` children of the context node other than the first `para` child of the context node
- `following-sibling::chapter[position()=1]` selects the next `chapter` sibling of the context node
- `preceding-sibling::chapter[position()=1]` selects the previous `chapter` sibling of the context node
- `/descendant::figure[position()=42]` selects the forty-second `figure` element in the document
- `/child::doc/child::chapter[position()=5]/child::section[position()=2]` selects the second `section` of the fifth `chapter` of the `doc` document element
- `child::para[attribute::type="warning"]` selects all `para` children of the context node that have a `type` attribute with value `warning`
- `child::para[attribute::type='warning'][position()=5]` selects the fifth `para` child of the context node that has a `type` attribute with value `warning`
- `child::para[position()=5][attribute::type="warning"]` selects the fifth `para` child of the context node if that child has a `type` attribute with value `warning`

- `child::chapter[child::title='Introduction']` selects the `chapter` children of the context node that have one or more `title` children with string-value equal to `Introduction`
- `child::chapter[child::title]` selects the `chapter` children of the context node that have one or more `title` children
- `child::*[self::chapter or self::appendix]` selects the `chapter` and `appendix` children of the context node
- `child::*[self::chapter or self::appendix][position()=last()]` selects the last `chapter` or `appendix` child of the context node

XPath: Exemples de location paths (Syntaxe simplifie)

- `para` selects the `para` element children of the context node
- `*` selects all element children of the context node
- `text()` selects all text node children of the context node
- `@name` selects the `name` attribute of the context node
- `@*` selects all the attributes of the context node
- `para[1]` selects the first `para` child of the context node
- `para[last()]` selects the last `para` child of the context node
- `*/para` selects all `para` grandchildren of the context node
- `/doc/chapter[5]/section[2]` selects the second `section` of the fifth `chapter` of the `doc`
- `chapter//para` selects the `para` element descendants of the `chapter` element children of the context node
- `//para` selects all the `para` descendants of the document root and thus selects all `para` elements in the same document as the context node
- `//olist/item` selects all the `item` elements in the same document as the context node that have an `olist` parent
- `.` selects the context node
- `.//para` selects the `para` element descendants of the context node
- `..` selects the parent of the context node
- `../@lang` selects the `lang` attribute of the parent of the context node
- `para[@type="warning"]` selects all `para` children of the context node that have a `type` attribute with value `warning`
- `para[@type="warning"][5]` selects the fifth `para` child of the context node that has a `type` attribute with value `warning`
- `para[5][@type="warning"]` selects the fifth `para` child of the context node if that child has a `type` attribute with value `warning`
- `chapter[title="Introduction"]` selects the `chapter` children of the context node that have one or more `title` children with string-value equal to `Introduction`
- `chapter[title]` selects the `chapter` children of the context node that have one or more `title` children
- `employee[@secretary and @assistant]` selects all the `employee` children of the context node that have both a `secretary` attribute and an `assistant` attribute

Exemple d'espaces de nomage concernant les éléments

Un exemple de préfixes qualifiant des nom d'éléments:

```
<x   xmlns:edi='http://ecommerce.org/schema'
     xmlns:doc='http://doc.org/schema'>
     <!-- le namespace de l'element"p" est: http://ecommerce.org/schema -->
   <edi:p unite='Euro'>32.18</edi:p>

      <!-- le namespace de l'element"p" est: http://doc.org/schema -->
   <doc:p id='premier'>mon paragraphe</doc:p>
 </x>
```

Exemple d'espaces de nomage par défaut :

Exemple 1: Le namespace par défaut

Un namespace par défaut s'applique a l'élément ou il est déclare et a tout ses éléments fils (sauf si un autre namespace est déclare).

```
<?xml version="1.0"?>
<!-- les elements sont dans le namespace HTML par defaut -->
<html xmlns='http://www.w3.org/TR/REC-html40'>
  <head><title>Frobnostication</title></head>
  <body><p>Moved to
    <a href='http://frob.com'>here</a>.</p></body>
</html>
```

Exemple 2: Namespace par défaut et namespace préfixé

Un namespace par défaut s'applique a l'élément ou il est déclare et a tout ses éléments fils (sauf si un autre namespace est déclare).

```
<?xml version="1.0"?>
<!-- les element sana prefixe sont dans le namespace "books" -->
<book xmlns='http://www.biblio.org/schema/books'
      xmlns:isbn='http://www.foo.org/ISBN'>
    <title>Cheaper by the Dozen</title>
    <isbn:number>1568491379</isbn:number>
</book>
```

Exemple 3: Multi-espaces de nomage par défaut :

```
<?xml version='1.0'?>
 <Beers xmlns='http://www.foo.org/biere/Schema'>
   <!-- le namespace par défaut est celui de HTML -->

   <table xmlns='http://www.w3.org/1999/xhtml'>
    <th><td>Name</td><td>Origin</td><td>Description</td></th>
    <tr>
       <td> <!-- le namespace par défaut est celui de biere -->
        <details xmlns='http://www.foo.org/biere/Schema'>
          <class>Bitter</class>
          <hop>Fuggles</hop>
          <pro>Wonderful hop, light alcohol, good summer beer</pro>
          <con>Fragile; excessive variance pub to pub</con>
          </details>
        </td>
      </tr>
    </table>
```

```
</Beers>
```

Exemple de multi-espaces de nomage

La déclaration du namespace s'applique a l'élément ou il est déclaré et a tout ses éléments fils (sauf si un autre namespace est déclaré).

Ex1:

```
<?xml version="1.0"?>
<!-- tous les éléments sont dans l'espace de nomage HTML  -->
<html:html xmlns:html='http://www.w3.org/TR/REC-html40'>
  <html:head><html:title>Frobnostication</html:title></html:head>
  <html:body><html:p>Moved to
    <html:a href='http://frob.com'>here.</html:a></html:p></html:body>
</html:html>
```

Ex2: Les préfixes de namespaces multiples sont déclarés comme attribut d'un élément:

```
<?xml version="1.0"?>
<!-- les deux préfixes de namespace sont disponibles -->
<bk:book xmlns:bk='http://www.biblio.org/schema/books'
         xmlns:isbn='http://www.foo.org/ISBN'>
    <bk:title>Cheaper by the Dozen</bk:title>
    <isbn:number>1568491379</isbn:number>
</bk:book>
```

Exemples: Portée du Namespace

Ex1: Namespaces par défaut et namespace avec préfixe

```
<?xml version="1.0"?>
<!-- le namespace par defaut est "books" -->
<book xmlns='http://www.biblio.org/schema/books'
      xmlns:isbn='http://www.foo.org/ISBN'>
    <title>Cheaper by the Dozen</title>
    <isbn:number>1568491379</isbn:number>
    <notes>
      <!--  HTML est le namespace par défaut pour ce commentaire -->
      <p xmlns='http://www.w3.org/TR/REC-html40'>
          This is a <i>funny</i> book!
      </p>
    </notes>
</book>
```

Ex2:les namespaces sont définis localement pour chaque élément concernlé

```
<?xml version="1.0"?>
<bk:books xmlns:bk="http://www.foo.org/book/schema">
 <bk:book>
  <bk:title>Inside Dynamic HTML</bk:title>
  <bk:author>
   <per:person name="Scott Isaacs" xmlns:per="http://www.foo.org/per/schema">
    <per:title>Program Manager</per:title>
   </per:person>
  </bk:author>
 </bk:book>
```

```
 <bk:book>
  <bk:title>ActiveX Controls Inside Out</bk:title>
  <bk:author>
   <per:person name="Adam Denning" xmlns:per="http://www.foo.org/per/schema">
    <per:title>Program Manager</per:title>
   </per:person>
  </bk:author>
 </bk:book>
</bk:books>
```

Ex3: même exemple que Ex 2 mais les namespaces sont définis dans l'élément racine

```
<?xml version="1.0"?>
<bk:books xmlns:bk="http://www.foo.org/book/schema"
          xmlns:per="http://www.foo.org/per/schema">
 <bk:book>
  <bk:title>Inside Dynamic HTML</bk:title>
  <bk:author>
   <per:person name="Scott Isaacs">
    <per:title>Program Manager</per:title>
   </per:person>
  </bk:author>
 </bk:book>
 <bk:book>
  <bk:title>ActiveX Controls Inside Out</bk:title>
  <bk:author>
   <per:person name="Adam Denning">
    <per:title>Program Manager</per:title>
   </per:person>
  </bk:author>
 </bk:book>
</bk:books>
```

Exemple d'utilisation de DOM HTML

```
<HTML>
<HEAD>
  <SCRIPT language ="javascript>
  function affiche()
{
document.open();
document.write('<HTML>');
document.write('<HEAD><TITLE>Exemple de DOM HTML</TITLE></HEAD>');
document.write('<BODY><H1>Exemple de page construite dynamiquement avec DOM
HTML</H1></BODY>');
document.write('</HTML>');
document.close();
}
</SCRIPT>
</HEAD>
<BODY>
 <SCRIPT> affiche();  </SCRIPT>
</BODY>
</HTML>
```

Résultat:

Exemple de page construite dynamiquement avec DOM HTML

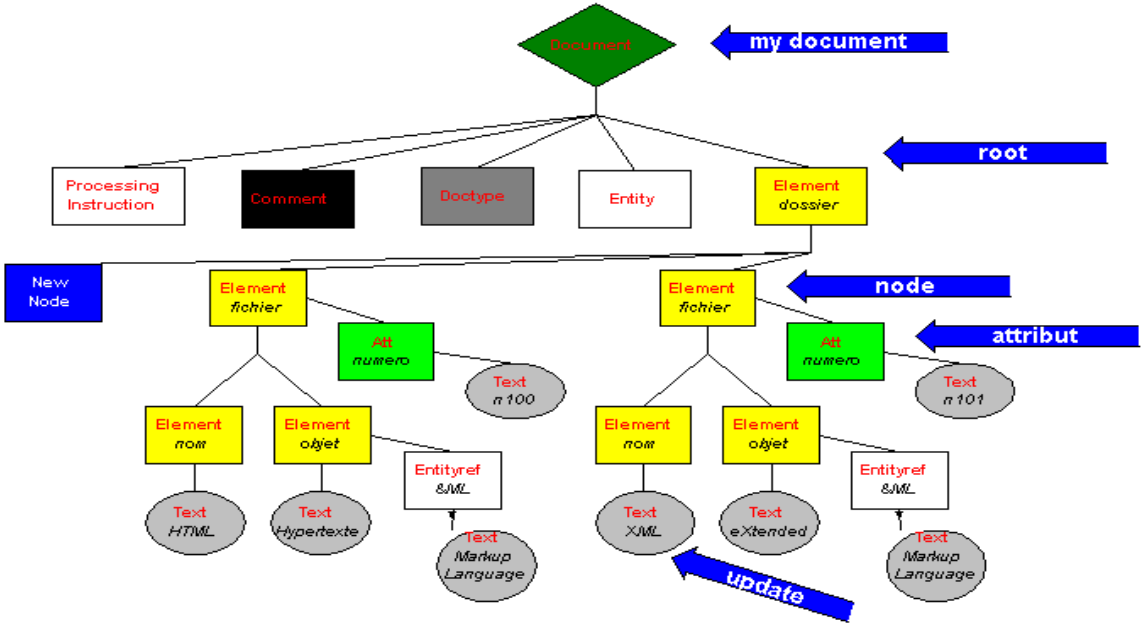
DOM:Navigation dans l'arbre

Exemple de propriétés et méthodes qui permettent d'accéder aux noeuds de l'arbre du document XML.

- **documentElement:** Renvoie l'élément racine du XML document.
- **childNodes:** Renvoi une liste de noeuds des fils de ce noeud
- **item:** Accéde individuellement aux noeuds de la liste a travers un Index de zéro a N - l'item(0) renvoie le premier noeud fils.
- **text:** Renvoi le contenu textuel du noeud.

Navigation:

Noeud	Méthodes/propriétés	Renvoi
Noeud Racine	xmldoc.documentElement	
Nom de l'élément Racine	xmldoc.documentElement.nodeName	dossier
Premier fils de l'élément racine	xmldoc.documentElement.childNodes.item(0)	fichier
Nom de l'attribut	xmldoc.documentElement.childNodes.item(0).attributes.item(0).nodeName	numero
Valeur de l'attribut	xmldoc.documentElement.childNodes.item(0).getAttribute('numero')	n100
Premier fils de l'élément	xmldoc.documentElement.childNodes.item(0).childNodes.item(0)	nom
contenu de l'élément	xmldoc.documentElement.childNodes.item(0).childNodes.item(0).text	HTML
Deuxième noeud fils de l'élément	xmldoc.documentElement.childNodes.item(0).childNodes.item(1)	objet
contenu de l'élément	xmldoc.documentElement.childNodes.item(0).childNodes.item(1).text	Hypertext Markup Language




```
root = mydocument.documentElement
node= root.childNodes.item(1)
attribut = node.attributes.getNamedItem("numero");
node.childNodes.item(0).childNodes.item(0).nodeValue = "XML";
root.insertBefore(NewNode,root.firstChild);
```

Ecrire un document XML, instance de cette DTD qui soit valide

1- La DTD

```
<!-- DTD de dossier -->

<!ENTITY ML "Markup Language">
<!ELEMENT dossier (fichier)+>
<!ELEMENT fichier (nom, objet)>
<!ATTLIST fichier numero ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT objet (#PCDATA)>
```

un exemple de document XML, instance de cette DTD qui soit valide

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

<!DOCTYPE dossier SYSTEM "dossier.dtd">

<dossier>

  <fichier numero="n100">
    <nom>HTML</nom>
    <objet>HyperTexte &ML;</objet>
  </fichier>

  <fichier numero="n101">
    <nom>XML</nom>
    <objet>eXtensible &ML;</objet>
  </fichier>

  <fichier numero="n102">
    <nom>XSL</nom>
    <objet>eXtensible Stylesheet language</objet>
  </fichier>

</dossier>
```

Ecrire une DTD pour le document XML ci dessous

le document XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE equipe SYSTEM "equipe.dtd">

<equipe>

 <salarie id="Big.Boss">
  <emploi><poste>Boss</poste> <type>Big</type></emploi>
    <email>chief@foo.com</email>
    <url href="www.foo.com/chief"/>
   <lien subordonnees="premier.ouvrier second.ouvrier trois.ouvrier quatre.ouvrier cinq.ouvrier"/>
 </salarie>

 <salarie id="premier.ouvrier">
  <emploi><poste>ouvrier</poste> <type>premier</type></emploi>
  <email>premier@foo.com</email>
  <email>premier@provider.com</email>
  <lien responsable="Big.Boss"/>
 </salarie>

 <salarie id="second.ouvrier">
  <emploi><poste>ouvrier</poste> <type>second</type></emploi>
  <email>second@foo.com</email>
  <lien responsable="Big.Boss"/>
 </salarie>

 <salarie id="trois.ouvrier">
  <emploi><poste>ouvrier</poste> <type>trois</type></emploi>
  <url href="www.foo.com/trois"/>
  <lien responsable="Big.Boss"/>
 </salarie>

 <salarie id="quatre.ouvrier">
  <emploi><poste>ouvrier</poste> <type>quatre</type></emploi>
  <email>quatre@foo.com</email>
    <url href="www.foo.com/quatre"/>
    <url href="www.provider.com/quatre"/>
    <lien responsable="Big.Boss"/>
  </salarie>

 <salarie id="cinq.ouvrier">
  <emploi><type>cinq</type><poste>ouvrier</poste> </emploi>
  <email>cinq@foo.com</email>
  <lien responsable="Big.Boss"/>
 </salarie>
```

```
</equipe>
```

la DTD

```
<!ELEMENT equipe (salarie)+>

<!ELEMENT salarie (emploi,email*,url*,lien?)>
  <!ATTLIST salarie id ID #REQUIRED>

<!ELEMENT emploi ((poste,type)|(type,poste))>
<!ELEMENT poste (#PCDATA)>
<!ELEMENT type  (#PCDATA)>

<!ELEMENT email (#PCDATA)>

<!ELEMENT url EMPTY>
   <!ATTLIST url href CDATA #REQUIRED>

<!ELEMENT lien EMPTY>
  <!ATTLIST lien responsable  IDREF  #IMPLIED>
  <!ATTLIST lien subordonnees IDREFS #IMPLIED>
```

Ex: Entités Internes/Externes Générales

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE book SYSTEM "book.dtd"
[
<!ENTITY toc SYSTEM "toc.xml">
<!ENTITY tm "Thierry MICHEL">
<!ENTITY chap1 SYSTEM "chapters/c1.xml">
<!ENTITY chap2 SYSTEM "chapters/c2.xml">
]>

<book>
   <head>&toc;</head>
     <body>
      <aut>auteur:&tm;</aut>
       &chap1;
       &chap2;
     </body>
</book>
```

Ex: Entités Paramètres

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE book SYSTEM "book.dtd"
[
<!ENTITY %rights "Tous droits réservés.">
<!ENTITY book "Cours XML, %rights;">
]>
```

Ex: Entités Paramètres et Générales

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE book SYSTEM "book.dtd"
[
<!ENTITY %pub "&#xc9;ditions W3C.">
<!ENTITY %rights "Tous droits réservés.">
<!ENTITY book "Cours XML, &#xa9; 1998 %pub; %rights;">
]>

<-- dans le document --->
<livre>&book;</livre>
```

Ce qui nous affiche:

Cours XML, © 1998 Éditions W3C.Tous droits réservés.

EX:DTD externe au document XML:

Le fichier XML

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

<!-- DTD externe au document XML -->
<!DOCTYPE dossier SYSTEM "dossier.dtd">


<dossier chemin ="c:\documentsXML">
  <fichier nom="exemple.xml">
    ce fichier contient un exemple pour &titre;
  </fichier>

  <fichier nom="livre.xml">
     ce fichier decrit la structure de document
  </fichier>

</dossier>
```

La DTD (fichier dossier.dtd):

```
<!-- DTD du document XML dossier -->

<!ENTITY titre "Construire une application XML">
<!ELEMENT dossier (fichier)*>
<!ATTLIST dossier chemin CDATA #REQUIRED>
<!ELEMENT fichier (#PCDATA)>
<!ATTLIST fichier nom CDATA #REQUIRED>
```

EX:DTD interne au document XML:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>

<!-- DTD incluse dans le document XML -->

<!DOCTYPE dossier [
<!ENTITY titre "Construire une application XML">
<!ELEMENT dossier (fichier)*>
<!ATTLIST dossier chemin CDATA #REQUIRED>
<!ELEMENT fichier (#PCDATA)>
<!ATTLIST fichier nom CDATA #REQUIRED>
]>

<dossier chemin ="c:\documentsXML">
   <fichier nom="exemple.xml">
     ce fichier contient un exemple pour &titre;
   </fichier>

    <fichier nom="livre.xml">
       ce fichier decrit la structure du document
    </fichier>

</dossier>
```

EX:DTD interne et externe au document XML:

Fichier XML:

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

<!-- DTD externe au document XML et partie DTD incluse dans le document XML-->

<!DOCTYPE dossier SYSTEM "externe.dtd"

[
<!ELEMENT fichier #PCDATA>
<!ATTLIST fichier nom CDATA #REQUIRED>
]>
<dossier chemin ="c:\documentsXML">
  <fichier nom="exemple.xml">
   ce fichier contient un exemple pour &titre;
 </fichier>

  <fichier nom="livre.xml"
     ce fichier decrit la structure de document
  </fichier>

</dossier>
```

Partie externe (externe.dtd)

```
<!-- DTD externe au document XML -->

<!ENTITY  titre "Construire une application XML">
<!ELEMENT dossier fichier*>
<!ATTLIST dossier chemin CDATA #REQUIRED>
```

Exercice Fichier XML: Les 10 Erreurs ?

```
<AUCTIONBLOCK>
  <ITEM>
     <TITLE>Vase and Stones</TITLE>
     <ARTIST>Linda Mann</artist>
     <DIMENSIONS dim=2D>20x30 inches</DIMENSIONS>
      <MATERIALS><ITALIC>Oil</MATERIALS></ITALIC>
      <DATE><YEAR>1996</YEAR>
     <DESCRIPTION/>
     <PREVIEW_SMALL SRC="images/burl-s.jpg" WIDTH="300" HEIGHT="194" ALT="Vase and
Stones">
       <!--The small image -->
     </PREVIEW_SMALL>
     <PREVIEW_LARGE SRC="images/burl.jpg" WIDTH="640" HEIGHT="413" ALT="Vase and
Stones">
      <BIDS>
        <1BID>
         <PRICE>3300</PRICE>
         <xmlTIME>9:19:32 AM</xmlTIME>
         <BIDDER>John</BIDDER>
         <TIMESTAMP>1315</TIMESTAMP>
         </1BID>
</AUCTIONBLOCK
<SECTION>
<AUCTIONBLOCK>

        <BID>
         <PRICE>3100</PRICE>
         <TIME>2:48:08 PM</TIME>
         <BIDDER>Chris</BIDDER>
         <TIMESTAMP>1307</TIMESTAMP>
        </BID>

        <MTIME>
          <TIMESTAMP>1315</TIMESTAMP>
           This is an example of mixed content
        </MTIME>
   </ITEM>
</AUCTIONBLOCK>
```

les 10 erreurs

```
<?xml version="1.0"?>
<AUCTIONBLOCK>
  <ITEM>
     <TITLE>Vase and Stones</TITLE>
     <ARTIST>Linda Mann</artist>
     <DIMENSIONS dim=2D >20x30 inches</DIMENSIONS>
     <MATERIALS><ITALIC>Oil</MATERIALS></ITALIC>
     <DATE><YEAR>1996</YEAR>
     <DESCRIPTION/>
     <PREVIEW_SMALL SRC="images/burl-s.jpg" WIDTH="300" HEIGHT="194" ALT="Vase and
Stones">
       <!--The small image -->
```

```
     </PREVIEW_SMALL>
     <PREVIEW_LARGE SRC="images/burl.jpg" WIDTH="640" HEIGHT="413" ALT="Vase and
Stones"/>
      <BIDS>
        <1BID>
         <PRICE>3300</PRICE>
         <xmlTIME>9:19:32 AM</xmlTIME>
         <BIDDER>John</BIDDER>
         <TIMESTAMP>1315</TIMESTAMP>
         </1BID>
</AUCTIONBLOCK>
<SECTION>
<AUCTIONBLOCK>
        <BID>
         <PRICE>3100</PRICE>
         <TIME>2:48:08 PM</TIME>
         <BIDDER>Chris</BIDDER>
         <TIMESTAMP>1307</TIMESTAMP>
        </BID>
       </BIDS>
        <MTIME>
          <TIMESTAMP>1315</TIMESTAMP>
           This is an example of mixed content
        </MTIME>
   </ITEM>
</AUCTIONBLOCK>
```

Fichier XML bien Forme

```
<?xml version="1.0"?>
<AUCTIONBLOCK>
  <ITEM>
     <TITLE>Vase and Stones</TITLE>
     <ARTIST>Linda Mann</ARTIST>
     <DIMENSIONS>20x30 inches</DIMENSIONS>
     <MATERIALS>Oil</MATERIALS>
     <YEAR>1996</YEAR>
     <DESCRIPTION/>
     <PREVIEW_SMALL SRC="images/burl-s.jpg" WIDTH="300" HEIGHT="194" ALT="Vase and
Stones">
       <!--The small image-->
     </PREVIEW_SMALL>
     <PREVIEW_LARGE SRC="images/burl.jpg" WIDTH="640" HEIGHT="413" ALT="Vase and
Stones"/>
      <BIDS>
        <BID>
         <PRICE>3300</PRICE>
         <TIME>9:19:32 AM</TIME>
         <BIDDER>John</BIDDER>
         <TIMESTAMP>1315</TIMESTAMP>
         </BID>
        <SECTION/>
        <BID>
         <PRICE>3100</PRICE>
         <TIME>2:48:08 PM</TIME>
         <BIDDER>Chris</BIDDER>
```

```
        <TIMESTAMP>1307</TIMESTAMP>
       </BID>
      </BIDS>
       <MTIME>
         <TIMESTAMP>1315</TIMESTAMP>
          This is an example of mixed content
       </MTIME>
  </ITEM>
</AUCTIONBLOCK>
```

Exemple de Cascades CSS

```
<HTML>
<HEAD>
<TITLE>Example de CSS</TITLE>
<link rel="stylesheet" type="text/css" href="StyleSheets/home.css" >

<STYLE type="text/css">
    BODY { font-family: arial;
           color: black;
           background-color: white;}
    P      {text-align: left;   color: blue; }
    .rouge {text-align: left;   color: red; }
    SPAN   {color: orange; }

    A:hover { background: #FFA }
</STYLE>
</HEAD>
<BODY>
 <H1>Example de Cascades</H1>
<P> Ce paragraphe est bleue</P>
<P class="rouge"> Ce paragraphe est rouge</P>
<P style="color:green"> Ce paragraphe est vert avec
<SPAN> une insertion orange</SPAN> fin </P>
<BODY><HTML>
```

Fichier affiche:

Ce paragraphe est bleue

Ce paragraphe est rouge

Ce paragraphe est vert avec une insertion orange fin.