

1. Introduction

1.1 Problématique des Systèmes d'Information

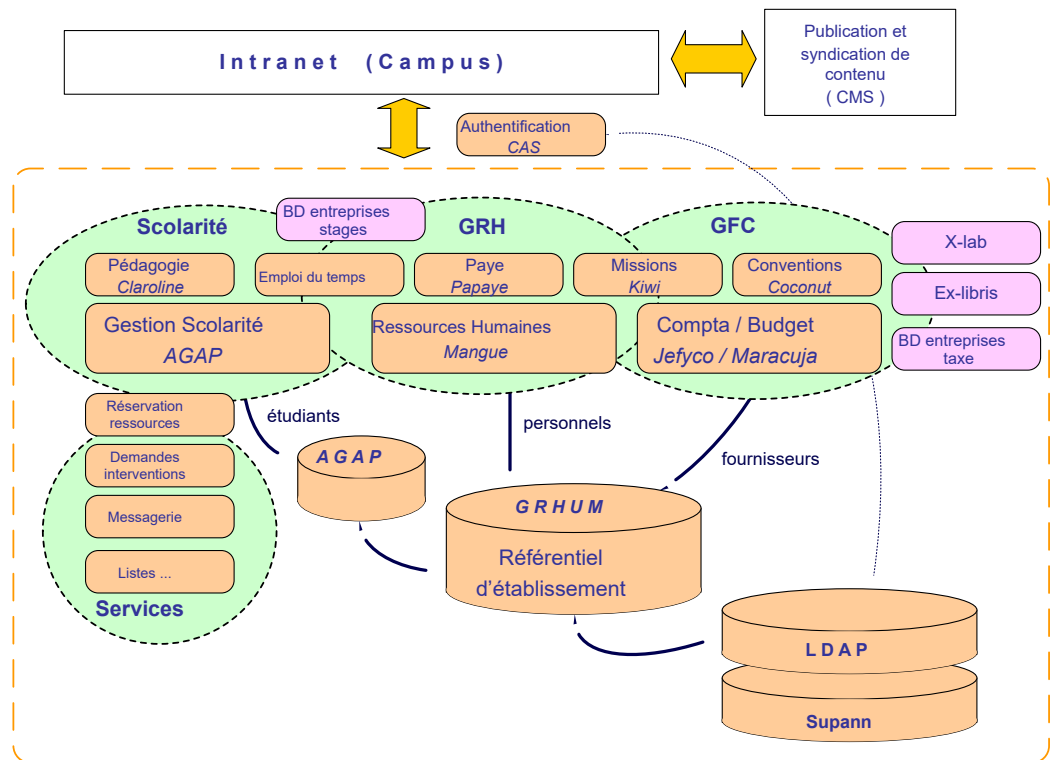
1.1.1 Qu'est-ce qu'un système d'information ?

D'après Wikipédia [http://fr.wikipedia.org/wiki/Syst%C3%A8me_d%27information], un **système d'information** (SI) représente l'ensemble des éléments participant à la gestion, au stockage, au traitement, au transport et à la diffusion de l'information au sein d'une organisation.

Cette définition englobe les éléments matériels (*serveurs, supports de stockage, ...*), logiciels (*applications, ...*), de communication (*réseau, ...*), les bases de données, les procédures (*workflow, ...*), voire le personnel.

Quelle que soit sa finalité (*production, gestion, ...*), un système d'information est donc un environnement **complexe** d'éléments en **interaction**.

Voici à titre d'exemple un schéma général du SI de l'Ecole Centrale fin 2006 :



Ce type de complexité est classique dans le domaine des SI d'entreprises.

1.1.2 Maîtriser la complexité

Au sein d'un SI se côtoient de nombreuses applications avec des finalités variées (*scolarité, gestion des ressources humaines, gestion financière et comptable, services*), qui communiquent entr'elles et partagent des données. Ces applications sont souvent basées sur des technologies diverses (*java, php, delphi, C, Cobol, ...*) et tirent leurs données de sources variées (*Oracle, Sybase, MySQL, LDAP, microservices...*).

Au niveau système la maîtrise de cette complexité repose sur les points suivants :

- les données sont fortement structurées,
- les données sont parfaitement identifiées (*cf. métadonnées*),
- les données sont toujours correctes (*vérifications d'intégrité*),
- les données ont une source unique (*transformation et non duplication*),
- les échanges se font via des formats publics (*architectures orientées services - SOA*).

On montrera dans la suite du cours comment **XML** et les technologies associées (*DTD, Schémas, Transformations, services Web*) répondent à cette problématique, et l'on comprendra dès lors pourquoi **XML** est omniprésent au cœur des **SI**.

1.1.3 Sommaire du cours

Voici donc les divers aspects qui seront abordés dans le cadre de ce cours, et comment ils se rattachent à la problématique générale des systèmes d'information :

- Introduction (*généralités, historique, tour d'horizon des standards et applications*),
- **XML**, éléments de syntaxe (*structuration des données*),
- **DTD**, schémas - validation (*intégrité des données*),
- **Namespaces** - espaces de nommage (*interopérabilité des applications*),
- **Xpath, Xquery** - recherche d'informations (*requêtage*),
- **XSLT** (*transformations*),
- **XML-RPC** - services Web (*échanges*),

1.2 Eléments d'histoire

1.2.1 A l'origine était SGML ...

SGML (*Standard Generalized Markup Language*) :

- est un langage créé chez IBM dans les années 70,
- ayant connu développement international,
- faisant l'objet du standard ISO 8879 en 1986.

SGML est :

- un langage sémantique et structurel à balises pour des documents "texte",
- destiné à la gestion de documentations techniques volumineuses (*plusieurs milliers de pages*).

SGML a entre autres donné lieu à des applications pour le gouvernement américain (*bibliothèque du congrès*), les militaires, le secteur aéronautique (*documentations techniques*).

1.2.2 De SGML à XML

> HTML : une application SGML

En 1992 Tim Berners-Lee invente le Web. **HTML** est conçu sous la forme d'une application **SGML**.

En 1995 la guerre des navigateurs fait rage (*HTML 2.0 - HTML 3.2*), sous la pression des utilisateurs et des concepteurs de navigateurs sans cesse à la recherche de nouvelles fonctionnalités.

> Une évolution de SGML pour le Web

Les évolutions incessantes de la syntaxe **HTML**, conduisent à rêver d'un langage qui permettrait d'inventer ses propres balises...

Malheureusement **SGML**, conçu dans les années 70 pour des applications batch, est tellement complexe et souvent redondant, qu'il n'est pas envisageable de pouvoir l'interpréter en temps réel au sein d'un navigateur.

En 1995, il n'existe aucune application implémentant l'ensemble des fonctionnalités de **SGML**.

En 1996 commencent les réflexions pour la définition d'une version simplifiée de **SGML** applicable au "temps réel" (*au sein d'un navigateur*).

Le produit de ces réflexions s'appellera **XML**.

> XML

Le développement s'est effectué avec les contraintes suivantes :

- **XML** doit pouvoir être facilement utilisé sur l'internet.
- **XML** doit pouvoir supporter des applications variées.
- **XML** doit être compatible avec **SGML**.
- Il doit être facile de développer des programmes qui traitent des documents **XML**.
- **XML** doit comporter un minimum d'éléments optionnels, idéalement zéro.
- Les documents **XML** doivent être lisibles et compréhensibles par des humains.
- Les spécifications de **XML** doivent être concises et précises.
- Les documents **XML** doivent être faciles à créer.
- La compacité des documents **XML** n'est pas d'une importance fondamentale.

Le premier draft **XML** a été présenté lors de la conférence **SGML 96** à Boston en **novembre 1996**.

La première recommandation (*XML 1.0*) a été émise en **février 1998**.

Les spécifications de **XML 1.0** ont subi plusieurs révisions. La cinquième date de **novembre 2008**.

🔗 Vers la recommandation XML 1.0 la plus récente en cours.

[\[http://www.w3.org/TR/REC-xml\]](http://www.w3.org/TR/REC-xml)

Sans remplacer la version 1.0 qui reste toujours recommandée, **XML 1.1** (*seconde version, septembre 2006*) prend en compte l'évolution du standard **Unicode** pour autoriser des caractères nouveaux pour les **noms XML** (*balises, attributs, ...*).

🔗 Vers la recommandation XML 1.1 la plus récente en cours.

[\[http://www.w3.org/TR/xml11\]](http://www.w3.org/TR/xml11)

1.2.3 En résumé

XML est une solution :

- qui s'appuie sur l'expérience **SGML**,
- constituant une simplification de **SGML**,
- faisant l'objet d'une recommandation [\[http://www.w3.org/TR/REC-xml\]](http://www.w3.org/TR/REC-xml) de la part du **W3C**,
- issue de la communauté "World-Wide-Web",
- enrichie depuis 1998 par de nombreuses extensions,
- suffisamment générique pour offrir des applications dans de nombreux domaines, qui n'ont souvent plus aucune relation directe avec le Web.

1.3 XML en 10 points

N.B. Le chapitre "XML en 10 points" a été élaboré d'après un document W3C.

[<http://www.w3.org/XML/1999/XML-in-10-points.fr.html>]

1.3.1 Une méthode pour structurer des données

XML n'est pas un langage ...

XML est un ensemble de règles pour la conception de formats texte permettant de structurer des données : *feuilles de calcul, carnets d'adresses, paramètres de configuration, transactions financières, dessins techniques...*

- permet la définition de formats non ambigus, extensibles, insensibles aux problèmes d'internationalisation/localisation,
- résout les problèmes de dépendance par rapport à certaines plates-formes,
- est conforme à Unicode.

1.3.2 XML ressemble à HTML

Comme **HTML**, **XML** identifie des éléments délimités par des balises de début et de fin, éventuellement affectés d'attributs :

```
<élément attribut="valeur">Contenu de l'élément</élément>
```

Toutefois :

- **HTML** possède un jeu de balises et d'attributs bien définis,
- avec **XML**, l'interprétation d'une balise dépend de l'application :

```
<article>
  <titre>
    Introduction à XML
  </titre>
  . . .
</article>
```

```
<personne>
  <titre>
    Altesse Sérénissime
  </titre>
  . . .
</personne>
```

1.3.3 XML : du texte pas forcément lu... par des humains

Un fichier **XML** est un fichier texte : l'édition et la correction sont donc possibles à l'aide d'un simple éditeur.

Ces opérations sont toutefois réservées à des experts car la syntaxe des applications est souvent très stricte (*bien plus que pour HTML*).

Les spécifications sont très explicites : une syntaxe approximative doit obligatoirement provoquer l'arrêt de l'application (*balise ouvrante non fermée, valeur d'un attribut sans guillemets, ...*).

1.3.4 XML est verbeux, mais ce n'est pas un problème

Un fichier **XML** n'est pas conçu pour optimiser l'espace de stockage (*fichier texte + balises*) :

Exemple :

[<http://dmolinarius.github.io/demofiles/mod-84/xml/intro/xml-10/point4-exemple.fr.html>]

Toutefois, cet aspect est pleinement assumé par les concepteurs.

En effet, les avantages du format texte sont évidents (*cf. point précédent*) et les inconvénients sont limités :

- l'espace disque est bon marché,
- il existe des possibilités de compression rapides et efficaces (*zip, gzip...*),
- certains protocoles de communication compressent à la volée (*modems, HTTP/1.1...*)

1.3.5 XML est une famille de technologies

XML définit ce qu'est une balise ou un attribut, mais ne possède pas de balises ni d'attributs en propre (*XML n'est pas un langage*).

XML a mis en place un cadre permettant le développement d'outils génériques et la spécification d'applications :

- parseurs, validateurs (*DTD, Schémas XML*), navigateurs (*CSS*),
- web (*XHTML*),
- moteurs de transformation XML vers XML (*XSL*), publication (*XSL-FO*),
- liens et sélecteurs (*XPath, XLink, XPointer*),
- protocoles de communication (*WebDAV, XMPP*), services Web (*XML-RPC, Soap*)...

1.3.6 XML est une technologie récente, mais éprouvée

Les premiers efforts de développement ont débuté en 1996.

Les premières spécifications datent de 1998.

Une technologie immature ?

Non, car basée sur l'expérience **SGML** (*années 70/80*) et **HTML** (*depuis 1990*).

1.3.7 XML fait passer HTML à XHTML

HTML 4.0 est une application **SGML**.

XHTML 1.0 est une application **XML**.

Les modifications syntaxiques sont mineures (*guillemets, balises fermantes, ...*).

Il est possible d'assurer la compatibilité avec les navigateurs pré-XML :

```

```

1.3.8 XML est modulaire

Un même document **XML** peut mêler des syntaxes provenant d'applications diverses :

en théorie un document web peut par exemple contenir à la fois des balises **XHTML**, des balises **MathML** (*formules mathématiques*), des balises **SVG** (*graphiques vectoriels*) et, pourquoi pas des balises **SMIL** (*animations*).

Les conflits potentiels entre les noms de balises et d'attributs provenant d'applications différentes sont gérés grâce aux espaces de noms (*XML Namespaces*).

1.3.9 XML est le fondement de RDF et du Web Sémantique

RDF (*Resource Description Framework*) est une recommandation du **W3C** pour les métadonnées au format **XML**.

Le **Web Sémantique** est une activité du **W3C** visant à permettre le développement d'applications réparties (*moteurs de recherche, e-commerce, annuaires...*) basées sur l'exploitation des métadonnées incluses au sein de documents **XML** disponibles en ligne.

1.3.10 XML est libre de droits, indépendant des plates-formes et correctement supporté

Les spécifications de **XML** sont publiques et libres de droit.

Chacune des technologies du **W3C** doit donner lieu à plusieurs implémentations indépendantes avant d'être recommandée (*i.e. publiée sous forme d'une recommandation*).

L'utilisation de technologies **XML** facilite le développement d'applications grâce à la disponibilité de modules logiciels standards, bien testés et maintenus, souvent eux-mêmes libres de droits (*parseurs, validateurs, moteurs de transformation, moteurs de recherche, éditeurs...*).

1.4 Un survol des standards

1.4.1 Définition de Type de Document

Dès l'origine, comme **SGML**, **XML** permet de spécifier la syntaxe d'une **application XML** (*nom des balises, des attributs, type de contenu, valeur par défaut..*) à l'aide d'une **DTD** (*Document Type Definition*).

La syntaxe d'une **DTD**, héritée de **SGML**, est particulière :

```
<!DOCTYPE classe [  
  <!ELEMENT classe (promo,etudiant+)>  
  <!ELEMENT etudiant (nom, prenom)>  
  <!ELEMENT nom (#PCDATA)>  
  <!ELEMENT prenom (#PCDATA)>  
  <!ELEMENT promo (#PCDATA)>  

```

Un **document XML** qui possède une **DTD** et dont le contenu est conforme à sa **DTD** est un **document XML valide** (*cf. sens anglo-saxon*).

Il existe des outils génériques permettant de **valider** (*i.e. vérifier la validité*) un document **XML** quelconque.

Exemple :

[\[http://dmolinarius.github.io/demofiles/mod-84/xml/std/dtd-validation.html\]](http://dmolinarius.github.io/demofiles/mod-84/xml/std/dtd-validation.html)

 Vers le cours sur les DTD

[\[http://dmolinarius.github.io/demofiles/mod-84/dtd.pdf\]](http://dmolinarius.github.io/demofiles/mod-84/dtd.pdf)

1.4.2 Espaces de Noms - XML Namespaces

 Site de référence

[\[http://www.w3.org/XML/Core/\]](http://www.w3.org/XML/Core/)

Les espaces de noms (*XML Namespaces*) ont très tôt fait l'objet d'une recommandation du **W3C** (*janvier 1999*). Ils permettent d'éviter les conflits potentiels entre les noms de balises et d'attributs lorsqu'on désire construire des documents mêlant des vocabulaires provenant d'applications différentes :

```
<article>
  <titre>
    Introduction à XML
  </titre>
  . . .
</article>
```

```
<personne>
  <titre>
    Altesse Sérénissime
  </titre>
  . . .
</personne>
```

Pour cela, le domaine de validité de chaque nom (*éléments et attributs*) est précisé grâce à un préfixe :

```
<publication:article>
  <publication:titre>Introduction à XML</publication:titre>
<publication:auteur>
  <personne:titre>Prof.</personne:titre>
  <personne:prenom>Daniel</personne:prenom>
  <personne:nom>Muller</personne:nom>
</publication:auteur>
</publication:article>
```

N.B. Ce n'est pas le préfixe qui identifie l'espace de noms, mais un **URI** associé au préfixe. Ces aspects seront vus dans le chapitre sur les espaces de noms.

Exemple :

[\[http://dmolinarius.github.io/demofiles/mod-84/xml/std/xmlns-exemple.html\]](http://dmolinarius.github.io/demofiles/mod-84/xml/std/xmlns-exemple.html)

 Vers le cours sur les espaces de noms

[\[http://dmolinarius.github.io/demofiles/mod-84/xmlns.pdf\]](http://dmolinarius.github.io/demofiles/mod-84/xmlns.pdf)

1.4.3 Schémas XML

 Site de référence

[\[http://www.w3.org/XML/Schema\]](http://www.w3.org/XML/Schema)

Espace de noms : <http://www.w3.org/2001/XMLSchema>

Préfixe courant : *xs*

Une **DTD** permet, aux fins de validation, de définir la syntaxe d'une application en indiquant le nom des balises et des attributs autorisés, le contexte dans lequel ils sont autorisés, ainsi que les valeurs autorisées pour certains attributs.

Toutefois, les **DTD** présentent plusieurs défauts, en effet : leur syntaxe est particulière (*non-XML, héritée de SGML*), elles décrivent uniquement la structure du document et non son contenu, et les données ne sont pas typées (*elles sont toutes considérées comme des chaînes de caractères*).

XML Schema est une **application XML** spécifiée par le **W3C** (*mai 2001*) qui comble ces lacunes.

Voici un extrait de schéma :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="page">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="titre" minOccurs="1" maxOccurs="1"/>
        <xs:group ref="contenuPage" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="id" type="pageID" use="required"/>
    </xs:complexType>
  </xs:element>
  . . .
</xs:schema>
```

Exemple détaillé :

[\[http://dmolinarius.github.io/demofiles/mod-84/xml/std/xsd-exemple.html\]](http://dmolinarius.github.io/demofiles/mod-84/xml/std/xsd-exemple.html)

📄 Vers le cours sur les schémas XML

[\[http://dmolinarius.github.io/demofiles/mod-84/xsd.pdf\]](http://dmolinarius.github.io/demofiles/mod-84/xsd.pdf)

1.4.4 Feuilles de style CSS

📄 Site de référence

[\[http://www.w3.org/Style/CSS/\]](http://www.w3.org/Style/CSS/)

Les navigateurs récents sont compatibles **XML**. Ceci signifie de prime abord qu'ils sont capables d'afficher un document en visualisant l'**arbre XML**.

Exemple :

[\[http://dmolinarius.github.io/demofiles/mod-84/xml/std/css.xml\]](http://dmolinarius.github.io/demofiles/mod-84/xml/std/css.xml)

Toutefois, contrairement au cas de **HTML**, la signification des éléments et des attributs leur est totalement étrangère.

Pour afficher un tel document de manière plus ergonomique qu'un arbre il est nécessaire de disposer d'une **feuille de style** indiquant quel doit être le format d'affichage de **chacun des éléments** du document.

Le langage **CSS** (*Cascading Style Sheets*) est un langage de feuilles de style, toujours en évolution à l'heure actuelle (2004), et qui a fait l'objet de plusieurs recommandations du **W3C** depuis 1996.

CSS est compatible **XML** et correctement reconnu par les navigateurs :

```
classe {
    display: block;
    padding: 8px;
    line-height: 1.33;
    background-color: #FFCC66;
    font-family: Arial, Helvetica, Sans-serif;
    color: black;
}
```

Si le document **XML** référence la feuille de style, il est affiché correctement par le navigateur :

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

Exemple :

[\[http://dmolinarius.github.io/demofiles/mod-84/xml/std/css-exemple.xml\]](http://dmolinarius.github.io/demofiles/mod-84/xml/std/css-exemple.xml)

Toutefois, les feuilles de style à mettre en oeuvre sont bien plus complexes que celles afférentes à un document **HTML** puisque contrairement à ce qui se passe pour ces dernières on ne peut pas se contenter de donner les écarts par rapport à un comportement par défaut du navigateur.

Les feuilles de style **CSS** qui s'appliquent à des documents **XML**, doivent préciser l'ensemble des caractéristiques permettant d'afficher chacun des éléments (*type, marges, couleurs, polices ...*).

A remarquer également que si le style est précisé par la feuille **CSS**, le contenu de éléments et leur ordre d'apparition à l'écran sont strictement imposés par le **document source**.

1.4.5 Feuilles de style XSL

📄 Site de référence

[\[http://www.w3.org/Style/XSL/\]](http://www.w3.org/Style/XSL/)


CSS possède des inconvénients : la syntaxe est très spécifique (*non-XML*) et surtout, il n'est pas possible de transformer le document (*changer l'ordre des éléments, choisir de ne pas en afficher certains, générer du contenu ...*).

Ces remarques ont conduit le **W3C** au développement d'un mécanisme de feuilles de style pour **XML** nommé **XSL** (*XML Style Sheets*).

XSL a été conçu en séparant ses deux fonctionnalités principales :

- **XSLT** (*XSL Transformation*), une **application XML** générique qui permet (*moyennant une feuille de style XSL*) de transformer tout **document XML** (*application x*) en un autre (*application y*),
- **XSL-FO** (*XSL Formatting Objects*), une **application XML** décrivant le contenu et l'aspect visuel d'un document, basée sur les fonctionnalités de **CSS-2** associées à une syntaxe **XML**.

1.4.6 Transformation de documents

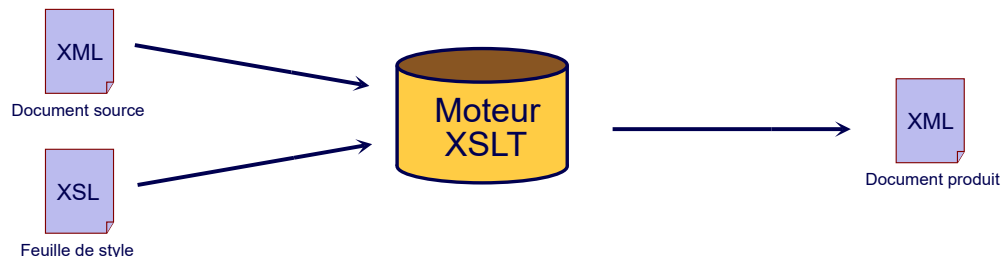
 Site de référence

[<http://www.w3.org/Style/XSL/>]

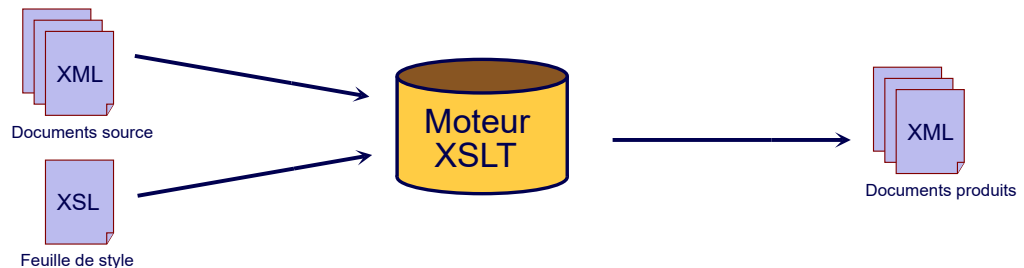
Espace de noms : <http://www.w3.org/1999/XSL/Transform>

Préfixe courant : *xsl*

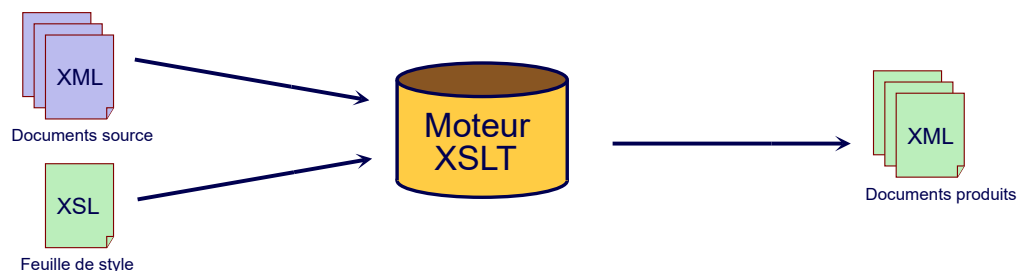
XSLT est une **application XML** recommandée par le **W3C** (*novembre 1999*) permettant de transformer un document **XML** en un autre document, **XML** ou non, à l'aide de règles spécifiées dans une feuille de style :



L'intérêt du procédé réside dans le fait qu'un nombre important de documents sources peut être transformé avec la même feuille de style, afin de produire, à des fins de publication, des documents possédant une identité visuelle commune :



Il suffit ensuite, sans toucher aux sources d'information, de modifier la feuille de style, pour modifier la présentation des documents produits :



1.4.7 XSL Formatting Objects

🔗 Site de référence

[\[http://www.w3.org/Style/XSL/\]](http://www.w3.org/Style/XSL/)

Espace de noms : <http://www.w3.org/1999/XSL/Format>

Préfixe courant : *fo*

XSL-FO est une **application XML** recommandée par le **W3C** (version 1.0 en octobre 2001, 1.1 en décembre 2006) basée sur les fonctionnalités de **CSS-2**, décrivant le contenu et l'aspect visuel d'un document :

```
<fo:root>
  <fo:layout-master-set>
    <fo:simple-page-master master-name="simple"
      margin-left="1.5cm" margin-right="1.5cm"
      margin-top="0.5cm" margin-bottom="0.5cm"
      page-width="21cm" page-height="29.7cm">
      . . .
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="simple">
    <fo:static-content flow-name="xsl-region-after"
      font-family="Arial" font-size="8pt">
      . . .
    <fo:block text-align="left"
      space-before="1.0em" space-after="1.5em"
      color="#000088" font-family="Arial"
      font-weight="bold" font-size="16pt">
      1. Publication de cours avec XML
    </fo:block>
    . . .
  </fo:static-content>
</fo:page-sequence>
</fo:root>
```

Un document **XSL-FO** comprend toutes les informations nécessaires à la visualisation ou à l'impression d'un document. Sémantiquement, **XSL-FO** peut de ce fait être comparé à des langages comme **Postscript** ou **pdf**.

Il existe malheureusement très peu d'environnements logiciels capables de rendre **XSL-FO** de manière native, (*navigateurs ?*), et il n'existe en particulier aucune imprimante acceptant directement ce langage.

Les applications existantes permettent en général de transformer un document **XSL-FO** en **pdf** (ou autres formats) aux fins d'impression (cf. *fo*), éventuellement après pré-visualisation sur écran (cf. *Antenna House XSL Formatter*).

1.4.8 Sélecteurs XPath

🔗 Site de référence

[\[http://www.w3.org/Style/XSL/\]](http://www.w3.org/Style/XSL/)

Pour désigner les éléments auxquels doit s'appliquer une certaine règle de mise en forme, **CSS** utilise des **sélecteurs**.

De manière similaire, **XSLT** nécessite un mécanisme pour désigner les éléments à transformer. Ce mécanisme, recommandé par le **W3C** (1.0 en novembre 1999, 2.0 en janvier 2007), s'appelle **XPath** :

```
<xsl:apply-templates
  select="child::node()[starts-with(name(),'menu')]" />
<xsl:apply-templates select="logo" />
<xsl:apply-templates
  select="contenu|contenu_large|special" />
```

XPath a été conçu sous la forme d'un langage générique pour désigner des ensembles de noeuds (*nodeset*) d'un **arbre XML**. Ce langage est également utilisé par d'autres applications comme **XPointer**.

1.4.9 Développement SAX / DOM

 Site de référence SAX

[\[http://www.saxproject.org/\]](http://www.saxproject.org/)

 Site de référence DOM

[\[http://www.w3.org/DOM/\]](http://www.w3.org/DOM/)

Le module logiciel permettant de lire un **document XML** s'appelle un **parseur**.

Un **parseur XML** vérifie obligatoirement le caractère bien formé d'un document (*well-formedness*). La plupart des parseurs sont également capables de valider un document par rapport à une **DTD** ou un **schéma**.

Il y a de nombreux parseurs disponibles (*dont certains en open-source*) avec lesquels il est possible de s'interfacer lorsque l'on développe un logiciel qui travaille avec des documents **XML**.

Il existe deux façons différentes de s'interfacer avec un parseur : approche "événements" (*API SAX - simple API for XML*) ou travail sur l'arbre en mémoire (*API DOM - Document Object Model*). **SAX** est un standard de fait, **DOM** une recommandation du **W3C** (*level 1 oct. 1998, nov. 2000, level 2 nov. 2000, level 3 avril 2004*).

N.B. Il existe des parseurs **XML** pour tous les langages de programmation courants **C/C++**, **Java**, **perl**, **python**, **php**...

1.4.10 Pointeurs avec XPointer

 Site de référence

[\[http://www.w3.org/XML/Linking\]](http://www.w3.org/XML/Linking)

XPointer est une recommandation du **W3C** (*mars 2003*) qui définit le langage à utiliser au sein d'un **identifiant de fragment** pour référencer les ressources du type **text/xml** ou **application/xml**.

Cette méthode permet de désigner des parties de document en se basant sur une expression **XPath** :

```
#xpointer(id('table-principale')/tr[2]/td[1])
```

XPointer est un mécanisme générique utilisé par de nombreuses applications **XML** comme **XLink**, **XInclude**, **RDF** ou **SOAP**.

1.4.11 Liens avec XLink

 Site de référence

[\[http://www.w3.org/XML/Linking\]](http://www.w3.org/XML/Linking)

Espace de noms : <http://www.w3.org/1999/xlink>

Préfixe courant : *xlink*

XLink est une recommandation du **W3C** (*juin 2001*) qui décrit les éléments à insérer au sein de **documents XML** afin de mettre en place des liens entre ressources.

XLink est basé sur une **syntaxe XML** et autorise de simples liens unidirectionnels comme en **HTML** (*cf. élément A*), ou des liens plus sophistiqués (*bidirectionnels, multiples...*).

```
<etudiants
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:type="simple"
  xlink:href="promo_2005.xml"
  xlink:role="http://tic01.tic.ec-lyon.fr/annuaire_promo.xml"
  xlink:title="liste d'etudiants"
  xlink:show="new"
  xlink:actuate="onRequest">
  la promo 2005
</etudiants>
```

1.4.12 Avertissement

Le survol de quelques standards associés à **XML** qui vient d'être proposé est largement non exhaustif.

En effet, outre le fait que les technologies présentées sont pour la plupart encore en évolution (*nouvelles versions plus évoluées en cours de gestation*), il existe bien d'autres applications recommandées par le **W3C**, par d'autres organismes (*cf. OASIS*), voire normalisés.

Pour en suivre les évolutions, il est bon de connaître quelques sites de référence :

🔗 XML au W3C

[\[https://www.w3.org/standards/xml/\]](https://www.w3.org/standards/xml/)

🔗 Le consortium OASIS

[\[http://www.oasis-open.org/\]](http://www.oasis-open.org/)

🔗 www.xml.org

[\[http://www.xml.org/\]](http://www.xml.org/)

1.5 Exemples d'applications

1.5.1 Format Graphique X3D

🔗 Site de référence

[\[http://www.web3d.org/x3d/\]](http://www.web3d.org/x3d/)

Espace de noms : non

X3D est une **application XML** faisant l'objet d'un **Standard ISO** pour la représentation interactive d'objets et de scènes 3D.

X3D est développé par le consortium **Web3D**.

🔗 Vers le site du consortium Web3D

[\[http://www.web3d.org\]](http://www.web3d.org)

Les outils existants sont nombreux et vont de l'environnement auteur aux applications de rendu.

🔗 Ressources X3D

[\[http://www.web3d.org/x3d/content/examples/X3dResources.html\]](http://www.web3d.org/x3d/content/examples/X3dResources.html)

Les domaines d'application de **X3D** vont de la CAO mécanique au médical en passant par l'enseignement, les systèmes géographiques (*y compris extra-terrestres*), et les tutoriels de maintenance ou de sécurité.

1.5.2 Format Graphique SVG

🔗 Site de référence

[\[http://www.w3.org/Graphics/SVG/\]](http://www.w3.org/Graphics/SVG/)

Espace de noms : http://www.w3.org/2000/svg

Préfixe courant : svg

SVG (*Scalable Vector Graphics*) est une **application XML** pour la description de **graphiques vectoriels** recommandée par le **W3C** (*SVG 1.0 09/2001 remplacé par SVG 1.1 01/2003, et WD SVG 1.2 04/2004*).

Historiquement, **SVG** a été fortement supporté par **Adobe** qui distribuait gratuitement un viewer sous forme de plugin pour la plupart des navigateurs.


A l'heure actuelle, les éléments **SVG** font partie intégrante de **HTML5** et sont nativement supportés par les navigateurs.

De nombreux outils sont par ailleurs disponibles pour **SVG** (*viewers dédiés, plugins navigateurs, viewers pour mobiles, éditeurs natifs, logiciels exportant SVG, logiciels de conversion de format, génération dynamique côté serveur*).

Exemple :

[<http://dmolinarius.github.io/demofiles/mod-84/svg/intro/slide3-exemple1.svg>]

1.5.3 Format Multimédia SMIL

 Site de référence

[<http://www.w3.org/AudioVideo/>]

Espace de noms : <http://www.w3.org/2001/SMIL20/Language>


Préfixe courant : *smil*

SMIL (*Simple Multimedia Integration Language*) est une **application XML** pour l'intégration et la synchronisation de sources multimédia interactives (*vidéo, son, graphiques animés, texte, ...*) recommandée par le **W3C** (*SMIL 1.0 06/1998 - SMIL 2.0 08/2001*).

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <body>
    <par dur="15s">
      
    </par>
  </body>
</smil>
```

SMIL est bien supporté par le marché et effectivement implémenté par de nombreux viewers (*Real Player G2, Internet Explorer, Adobe SVG Viewer, Apple Quicktime, ...*)

1.5.4 Format d'échange WebDAV

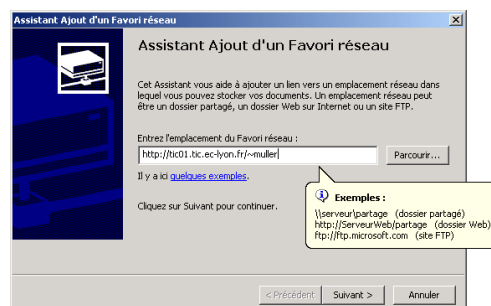
 Site de référence (au 17/10/2017)

[<https://web.archive.org/web/20171017202434/http://webdav.org:80/>]

Espace de noms : *DAV:*

Préfixe courant : *D*

WebDAV est l'un des protocoles implémentés par l'outil "favoris réseau" de Microsoft Windows.



WebDAV (*Web-based Distributed Authoring and Versioning*) est une extension du protocole **HTTP** qui permet d'éditer et de maintenir un jeu de documents situé sur un serveur Web distant (*RFC 2518 - 02/1999*).

WebDAV s'appuie sur **XML** pour bénéficier de l'extensibilité qu'il procure, et le support des jeux de caractères **ISO 10646 (I18N)**.

> Exemple de requête

```
PROPFIND /~muller/ HTTP/1.1
Host: tic01.tic.ec-lyon.fr
Depth: 1
Content-Type: text/xml; charset="utf-8"
Content-Length: 98
<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:allprop/>
</D:propfind>
```

> Exemple de réponse

(noter au passage l'usage intensif des espaces de noms ...)

```
HTTP/1.1 207 Multi-Status
Date: Fri, 22 Nov 2002 13:05:43 GMT
Server: Apache/1.3.27 (Unix) PHP/4.2.3 DAV/1.0.3 mod_ssl/2.8.11 OpenSSL/0.9.6g
Transfer-Encoding: chunked
Content-Type: text/xml; charset="utf-8"
ec5
<?xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:">
  <D:response xmlns:lp0="DAV:" xmlns:lp1="http://apache.org/dav/props/">
    <D:href>/~muller/images/</D:href>
    <D:propstat>
      <D:prop>
        <lp0:creationdate b:dt="dateTime.tz"
          xmlns:b="urn:uuid:c2f41010-65b3-11d1-a29f-00aa00c14882/">
          2002-10-09T10:58:02Z
        </lp0:creationdate>
        <lp0:getlastmodified b:dt="dateTime.rfc1123"
          xmlns:b="urn:uuid:c2f41010-65b3-11d1-a29f-00aa00c14882/">
          Wed, 09 Oct 2002 10:55:49 GMT
        </lp0:getlastmodified>
        <lp0:getetag>"32d61-1000-3da40b35"</lp0:getetag>
        <D:resourcetype><D:collection/></D:resourcetype>
        <D:getcontenttype>httpd/unix-directory</D:getcontenttype>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
  . . .
</D:multistatus>
```

1.5.5 Format d'échange XML RPC

 Site de référence

[<http://www.xmlrpc.com>]

XML RPC (*Remote Procedure Call*) est un mécanisme qui permet l'exécution de procédures distantes sur plates-formes hétérogènes via l'Internet, initié par Userland.

XML RPC s'appuie sur **HTTP** (pour le transport) et **XML** (pour le codage des données), et ouvre la porte aux **Web services**...

> Exemple de requête

```
POST /~muller/cours/xml/appl/xmlrpc-server.php HTTP/1.0
Host: tic01.tic.ec-lyon.fr
User-Agent: xmlrpc-client.php
Content-Type: text/xml
Content-Length: 153
Connection: Close
<?xml version="1.0"?>
<methodCall>
  <methodName>NomDepartement</methodName>
  <params>
    <param>
      <value>69</value>
    </param>
  </params>
</methodCall>
```

> Exemple de réponse

```
HTTP/1.1 200 OK
Date: Tue, 14 Sep 2004 21:34:38 GMT
Server: Apache/1.3.28
X-Powered-By: PHP/4.3.3
Connection: close
Content-Length: 170
Content-Type: text/xml
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <string>Rh&#xF4;ne</string>
      </value>
    </param>
  </params>
</methodResponse>
```

N.B. A l'heure actuelle, **XML RPC** a perdu de l'intérêt face à d'autres formats comme **JSON**.

1.5.6 Le Consortium OASIS

 Site de référence

[\[http://www.oasis-open.org\]](http://www.oasis-open.org)

OASIS (*Organization for the Advancement of Structured Information Standards*) est un "consortium global" à but non lucratif pour le développement, la convergence et l'adoption de standards e-business issu en 1998 de **SGML Open**, lui-même fondé en 1993.


 Vers la liste des membres

[\[http://www.oasis-open.org/about/\]](http://www.oasis-open.org/about/)

OASIS opère **www.xml.org** (référence pour les applications de XML & Web-services) ainsi que le site **xml.coverpages.org** (ressources en ligne sur les langages à balises).

 Vers le site **www.xml.org**

[\[http://www.xml.org\]](http://www.xml.org)

 Vers le site **xml.coverpages.org**


[\[http://xml.coverpages.org\]](http://xml.coverpages.org)

Voici quelques exemples de **Comités Techniques OASIS** : e-Government, Election and Voter Services, HumanMarkup, LegalXML,...

1.6 XHTML

1.6.1 Une reformulation de HTML 4.0 en XML 1.0

La spécification **XHTML 1.0** est une recommandation du **W3C** (janvier 2000 rev. août 2002).

 En savoir plus

[\[http://www.w3.org/TR/xhtml1/\]](http://www.w3.org/TR/xhtml1/)

Cette spécification définit **XHTML 1.0** comme une reformulation de **HTML 4** en une application **XML 1.0**, avec trois **DTD** (*Définitions de Type de Document*) correspondant à celles définies par **HTML 4** (*Strict, Transitional, Frameset*).

La compatibilité de **XHTML 1.0** avec les agents utilisateurs **HTML** (*navigateurs*) actuels ou plus anciens, est possible en suivant un ensemble raisonnable de règles.

1.6.2 Différences avec HTML 4.0

Un document **XHTML** doit être bien formé au sens de **XML** : tous les éléments doivent être correctement emboîtés.

XML et donc **XHTML** sont sensibles à la casse. Par convention les noms d'éléments et d'attributs **XHTML** sont en casse minuscule.

```
<html>
  <head>
    ...
  </head>
  <body>
    ... les noms d'éléments et d'attributs sont en minuscule ...
  </body>
</html>
```

La balise de fin est obligatoire.

```
<p>
  Les balises de fin sont obligatoires...
  même pour les éléments vides.
</p>
</img>
```

Les valeurs d'attributs doivent toujours être entre guillemets.

```
<table border="0">
  ...
</table>
```

En **HTML** certains attributs (*dits 'minimisés'*) ne prennent pas de valeur (*cf. CHECKED et NOWRAP*). **XML** ne supporte pas la minimisation de l'attribut.

```
<td nowrap="nowrap">
  Les attributs minimisés prennent leur nom pour valeur
</td>
```

La balise de fin des éléments vides peut être remplacée par un raccourci typographique. Les deux notations suivantes sont strictement équivalentes :

```
</img>

```

1.6.3 Problème des éléments **SCRIPT** et **STYLE**

En **XHTML**, les éléments **script** et **style** sont déclarés comme ayant un contenu de type **#PCDATA**, c'est à dire que le contenu des ces éléments est analysé par le client. Par suite, les caractères **<** et **&** seront traités comme le début d'un balisage.

Une première solution consiste à protéger ces caractères par des appels d'entité comme **<** ou **&** :

```
<script type="text/javascript">
  for ( i=0; i &lt; 10; i++ ) { ... }
</script>
```

La solution **XML** consisterait à emballer le contenu des éléments **script** à l'intérieur d'une section marquée **CDATA** ce qui évite l'interprétation de ces entités par les parseurs :

```
<script type="text/javascript">
  <![CDATA[
    ... Contenu de la script ...
  ]]>
</script>
```

Toutefois, si elle convient du point de vue de **XML**, cette solution pose problème car la déclaration **CDATA** n'est pas du code **Javascript** correct.

Démonstration et solution :

[\[http://dmolinarius.github.io/demofiles/mod-84/html/xhtml1/slide3-exemple1.html\]](http://dmolinarius.github.io/demofiles/mod-84/html/xhtml1/slide3-exemple1.html)

1.6.4 Problème des attributs **NAME** et **ID**

HTML 4 a défini l'attribut **name** (cf. éléments *A*, *applet*, *form*, *frame*, *iframe*, *img*, et *map*). De plus, **HTML 4** a également introduit l'attribut **id** qui peut s'appliquer entre autres à ces éléments-là.

En **XML**, les identificateurs partiels sont de type **ID**, et il ne peut y avoir qu'un seul identifiant de ce type par élément.

Un document **XHTML** bien formé doit utiliser de préférence l'attribut **id** et non pas **name** pour identifier les éléments listés ci-dessus.

```

```

Le cas des champs de formulaire (*input*, *select*, *textarea*...) est particulier puisque **id** et **name** ont un sens différent. Ces éléments peuvent porter les deux attributs, mais **XHTML** impose que leur valeur soit identique :

```
<input id="nom" name="nom" type="text"></input>
```