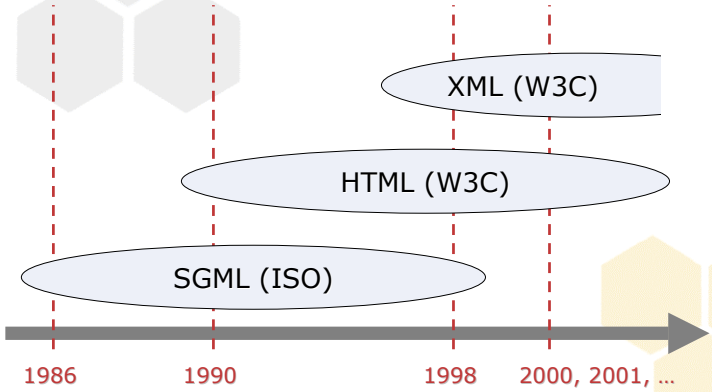
Historique

15 années d'expérience



x

Qu'est-ce que c'est ?

- **XML, eXtensible Markup Language**
- **XML 1.0 recommandation du W3C (10 février 1998)**
- **Sous-ensemble de SGML**
  - Langage à balises
  - 80% des fonctionnalités de SGML, 20% de sa complexité
- **Méta-langage**
  - Balises personnalisées
- **Séparer le contenu de la structure**
- **Séparer le contenu de la présentation**
- **Indépendant de toutes plate-formes et de tous langages**

x

Principes

1. **XML devra pouvoir être utilisé sans difficulté sur Internet**
2. **XML devra supporter une grande variété d'applications**
3. **XML devra être compatible avec SGML**
4. **Il devra être facile d'écrire des programmes traitant les documents XML**
5. **Le nombre d'options dans XML doit être réduit au minimum**
6. **Les documents XML devraient être lisible par l'homme**
7. **La conception de XML devraient être préparée rapidement**
8. **La conception de XML sera formelle et concise**
9. **Il devra être facile de créer des documents XML**
10. **La concision dans le balisage de XML a peu d'importance**

x

Exemple : Un compte bancaire

- **Compte bancaire**
  - numéro
  - client
  - solde
  - opérations
- **Avant XML**
  - Fichiers binaires
  - Fichiers texte

```
001
123456ZT 1542.6 06/12/2002 -200
```

x

Exemple : un compte bancaire

Avec XML

```
<?xml version="1.0" encoding="UTF-8"?>
<compte monnaie="euro">
        <numéro>123456ZT</numéro>
        <client>
                <nom>Dupont</nom>
                <prénom>Michel</prénom>
        </client>
        <solde>1542.6</solde>
        <opération id="789456123">
                <date format="iso-8601">2002-12-06</date>
                <montant>-200</montant>
        </opération>
</compte>
```


Contenu, Structure et Présentation

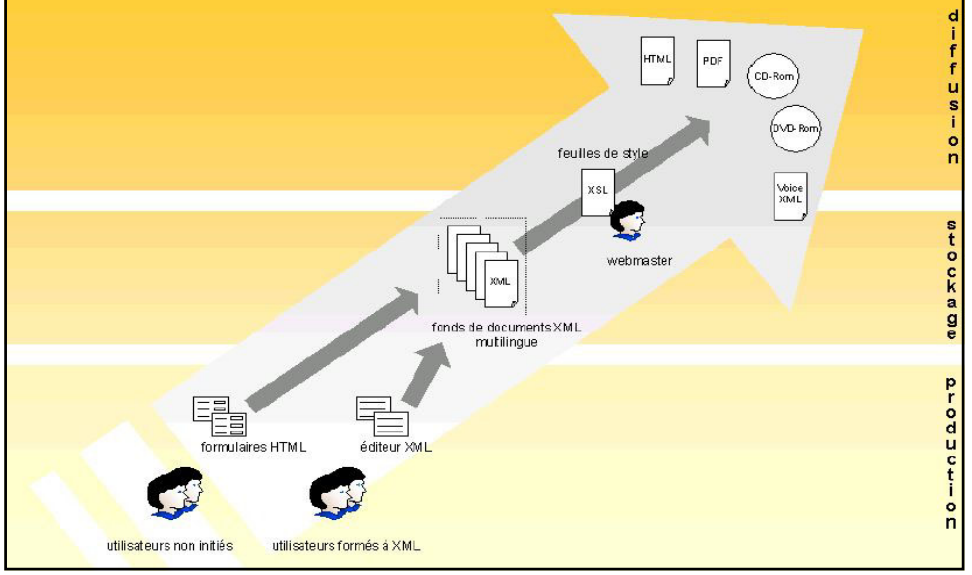
- **Document XML : Contenu**
  - balises : pas de signification prédéfinie
  - pas de présentation prédéfinie
- **Grammaire de document :**
  - DTD : Document Type Definition
  - ou XML-Schéma
- **Présentation et/ou traitement : feuilles de style**
  - CSS : Cascading Style Sheets
  - XSL : eXtensible Style Language

Pour se situer un peu ...

Analyse et Conception	Édition et production	Transformation et style	Diffusion et Consultation
DTD ou XSchéma	**XML**	**XSL XSLT/FO**	**XHTML PDF**
Structures	Données	Traitement des données	Documents utilisateurs
1	2	3	4

x

Production et Diffusion

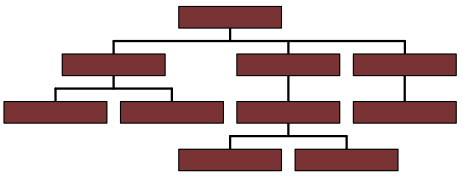


Syntaxe XML

- **XML utilise des balises pour délimiter les composants**
- **L'unité de base est un élément**

<nom> Patrice Bonhomme </nom>

- **Un élément peut contenir du texte et d'autres éléments**
  - Structure arborescente et hiérarchisation des éléments



Structure logique

- **Structure de document XML**
  - Un prologue
  - Un seul élément racine
  - Un arbre d'éléments (et leurs attributs)
  - Des commentaires

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- un exemple de compte -->
<compte monnaie="euro">
        <numéro>123456ZT</numéro>
        <client>
                <nom>Dupont</nom>
                <prénom>Michel</prénom>
        </client>
        <solde>1542.6</solde>
</compte>
```

Le prologue

- **Déclaration de document XML (recommandée)**

```
<?xml version="1.0"?>
```

- **Option**
  - Encodage des caractères

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

    - Valeur par défaut : **Unicode UTF-8**
  - Déclaration standalone

```
<?xml version="1.0" standalone="yes"?>
```

    - Stipulant si la DTD est autonome ou non, si toutes les déclarations concernant ce document XML sont stipulés dans le `!DOCTYPE`
  - Instruction de traitement

```
<?xml:stylesheet type="text/xsl" href="monXsl.xsl"?>
```

  - Déclaration de type de document

```
<!DOCTYPE membre SYSTEM "membre.dtd">
```

x

Autres éléments ou objets

- **Déclaration XML**

```
<?xml version="1.0"?>
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

- **Commentaires**

```
<!-- ceci est un commentaire -->
```

- **Section CDATA**

```
<![CDATA[Titi & gros minet]]>
```

Titi & gros minet

- **Instruction de traitement (pour une application)**

```
<?xml:stylesheet type="text/xsl" href="myXsl.xsl"?>
```

- **Entités : &entité;**

<	>	&	'	"
<	>	&	'	&quote;

x

Quelques propriétés de XML

- **Supporte les jeux de caractères `Unicode`**
  - Casse dépendant (minuscule ≠ majuscule)
  - Les données peuvent contenir presque tous les caractères
  - `Unicode` à tous les niveaux
- **Les espaces en dehors du balisage sont par défaut, préservés**
- **Les chaînes littérales sont toujours délimitées par des quotes simples ou doubles, par exemple :**

```
<!DOCTYPE DB SYSTEM "http://…/DB.dtd">
<MEMBRE TYPE="IE" ID="M28">
```

x

Formations lucid'i.t.

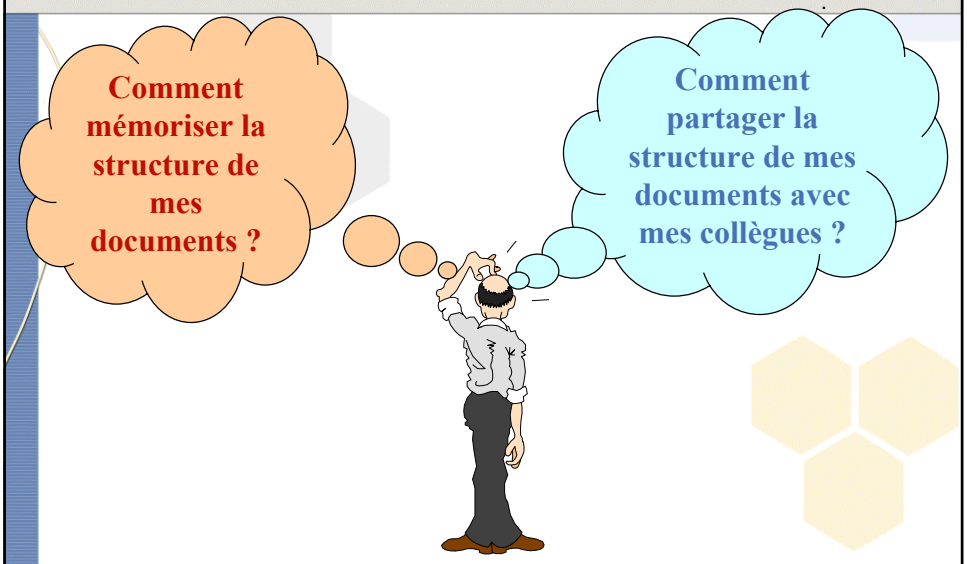
Grammaires et schémas

DTD

Schéma XML

Espaces de noms

Besoin d'une ontologie



x

Bien formé / Valide

- **Document bien formé**
  - Un seul élément racine
  - Balise correctement imbriquées : balise ouvrantes ont une balise fermante associée et il n'y a pas de chevauchement …
  - Le nom des balises est libre mais contient au moins une lettre en début
  - Les attributs des balises ont obligatoirement une valeur qui doit apparaître entre double ou simple quotes.
- **Document valide**
  - Associé à une définition de type de document et qu'il la respecte
    - Noms des éléments
    - Type
    - Répétition et ordre d'apparition

x

Document Type Definition

- **Définition de type de document**
  - Contraintes sur les noms des éléments et des attributs
  - Occurrences des éléments et des attributs
  - Structure et organisation des éléments
- **Approche SGML traditionnelle mais :**
  - Syntaxe simplifiée
  - Optionnelle en XML
    - Production valide et distribution bien-formée

Exemple de DTD

```
<!ELEMENT MEMBRE
          (LOGIN, NOM?, PRENOM?,MEL, TEL+, FAX*, EQUIPE)>
<!ELEMENT LOGIN EMPTY>
<!ATTLIST LOGIN id ID #REQUIRED>


<!ELEMENT NOM (#PCDATA)>
...
<!ENTITY RDP "Recherche et Développement Produits">
<!ENTITY eacute  "&#233;">
<!ENTITY chap1 SYSTEM "http://…/chapitre-1.xml">
...
```

Définition d'élément

```
<!ELEMENT nom-de-balise modèle-de-contenu>
```

- **Modèle de contenu**
  - Élément vide `<!ELEMENT soc EMPTY>`
  - Élément libre `<!ELEMENT post-scriptum ANY>`
  - Élément (texte) `<!ELEMENT nom (#PCDATA)>`
  - Élément fils `<!ELEMENT carnet (personne|société)>`
  - Contenu mixte `<!ELEMENT adresse (#PCDATA|cp|ville)*>`

x

Modèle de contenu

- **Pour écrire les contenus mixtes, on définit une expression de composition des éléments fils**
- **Opération de composition**
  - Séquence (,) `(nom, prenom, adresse)`
  - Alternative (|) `(personne | société)`
  - Occurrence
    - * : 0 à n occurrences
    - + : au moins 1 occurrences `(adresse, ligne-adr*)+`
    - ? : 0 ou 1 occurrence
  - Regroupement de fragment d'expression ()
    - `(nom, prenom?, email?)`

x

Définition d'attributs

```
<!ATTLIST    nom-balise nom-attribut
             modèle-de-contenu type requis>
```

```
<!ATTLIST société idname ID #REQUIRED
                type (SA|SARL|EURL) 'SA'>
```

- **CDATA** chaîne de caractère littérale

```
<!ATTLIST soc name CDATA #IMPLIED>
```

- **ID** identifiant dans le document

```
<!ATTLIST soc idname ID #REQUIRED>
```

- **IDREF, IDREFS** renvoi(s) vers ID à l'intérieur du doc

```
<!ATTLIST soc banque IDREF #IMPLIED>
```

- **ENUMERE** l'ensemble des valeurs possibles de l'attribut est défini.

```
<!ATTLIST personne fonction (ing|com|tech|admin) "ing">
```

x

Définition d'attributs

- **ENTITY, ENTITIES référence à une ou plusieurs entités externe non XML**

```
<!NOTATION gif SYSTEM "C:\Program\ACDSee32.exe">
<!ENTITY LigneBleu SYSTEM "logo-auf.gif" NDATA gif>
<!ELEMENT separateur EMPTY>
<!ATTLIST separateur img ENTITY #REQUIRED>
```

- **NMTOKEN, NMTOKENS nom symbolique (constitué que de caractères autorisés par XML)**

```
<!ATTLIST Foo FooToken NMTOKEN "xml">
Valide :
  <Foo FooToken="_17"/>
Non valide :
  <Foo FooToken="rouge&amp;bleu tok"/>
```

x

Valeur par défaut des attributs

- Valeur par défaut de l'attribut
- **#REQUIRED** l'attribut est obligatoire dans toutes les balises
- **#IMPLIED** l'attribut est optionnel
- **#FIXED 'valeur' :** la valeur de l'attribut est fixée, l'attribut est implicite pour toutes les balises pour lequel il est déclaré

```
<!ATTLIST personne fonction (ing|com|tech|admin)"ing">
<!ATTLIST soc idname ID #REQUIRED>
<!ATTLIST soc name CDATA #IMPLIED>
<!ATTLIST soc pays CDATA #FIXED "France">
```

x

Utilisation d'une DTD interne

```
<?xml version="1.0"?>

<!DOCTYPE MEMBRE [

<!ELEMENT MEMBRE (LOGIN, NOM?, PRENOM?,MEL, TEL+, FAX*, EQUIPE)>

<!ELEMENT LOGIN EMPTY>

<!ATTLIST LOGIN id ID #REQUIRED>

<!ELEMENT NOM (#PCDATA)>

…

]>

<MEMBRE TYPE="IE" ID="M28">

 …

</MEMBRE>
```

x

Utilisation d'une DTD externe

```
<!DOCTYPE MEMBRE SYSTEM "http://…/MEMBRE.dtd">

<MEMBRE TYPE="IE" ID="M28">

 …

</MEMBRE>
```

```
<!DOCTYPE MEMBRE SYSTEM "http://…/MEMBRE.dtd" [
<!ENTITY email  "@lucid-it.com">
]>

<MEMBRE TYPE="IE" ID="M28">
 …
        <mail>webmaster&email;</mail>
 …
</MEMBRE>
```


Utilisations d'une DTD (amont)

- **Une DTD permet de définir :**
  - Les éléments et leurs attributs (si nécessaire)
  - L'imbrication des éléments
  - Le contenu des éléments (texte, éléments, mixe, vide ou any)
  - Les séquences et l'ordre des éléments
  - Les choix : éléments/attributs optionnels ou obligatoire
  - Les occurrences multiples des éléments
  - Les types des attributs mais à un niveau basique
  - La modularité des DTD mais avec des contraintes d'ordre (si nécessaire)
  - Les entités et les notations (si nécessaire)

Utilisations d'une DTD (aval)

- **Si un document XML fait référence à une DTD, un *parser* XML validant détecte :**
  - Les éléments et attributs manquant
  - Les noms d'élément ou d'attribut erronés ou non définis dans la DTD
  - Les mauvaises imbrications d'éléments / les mauvais contenus
  - L'ordonnancement incorrect des éléments
  - Les mauvaises valeurs d'attributs (depuis les listes de valeurs)
  - les valeurs par défaut ou fixes des attributs

x

Entités

- **Servent à référencer d'autre objets**
- **Raccourcis syntaxique (remplacement)**
  - Dans la DTD (entités paramètre)
  - Dans le corps de document (entités interne, caractère)
- **Liaison d'objet externes :**
  - Objet XML (sous-document)
  - Objet non XML (notation, référence à des images, du contenu multimédia, etc)

x

Entités internes

- **Entités de caractères**
  - Représenter les caractères spéciaux

```
<entry>salaire &lt; 10K€</entry>
```

- **Entités générales internes**
  - But : nommer des expressions pour les réutiliser
  - Déclarée dans la DTD
  - Utilisée dans le document

```
<!ENTITY mail "@lucid.com">
<email>webmaster&mail;</email>
```

- **Entités paramètres**
  - Idem pour les DTDs
  - Déclarée dans la DTD
  - Utiliser dans la DTD

```
<!ENTITY % idt "(nom,prenom,age?)">
<!ELEMENT personne (%idt;,email,soc)>
```

x

Entités externes

- **Entités XML externes**
  - Référence des documents xml externes

```
<!ENTITY moncv SYSTEM "c:\moncv.xml">
```

- **Entités générales externes**
  - Référence des documents non XML (binaires)

```
<!ENTITY myPicture SYSTEM "c:\photo.jpg" NDATA jpg>
```

- **Notation**
  - Déclaration d'un type de fichier et de l'application associée

```
<!NOTATION jpg SYSTEM "C:\Program\ACDSee32.exe">
```

x

Identification de ressources externes

- **Une entité externe peut être identifiée :**
  - Par une URL
    - Ressources privés, mot clé SYSTEM

```
<!ENTITY maphoto SYSTEM " c:\photo.jpg ">
<!ENTITY mapage SYSTEM "http://www.lucid-it.com">
```

  - Par une FPI (Formal Public Identifier)
    - Identifiant public, ressources partagées, mot clé PUBLIC

```
<!ENTITY rec-XML PUBLIC "-//W3C//DOCUMENT Recommandation
open - The recommandation for XML 1.0//EN"
"http://www.w3.org/TR/1998/REC-xml-19980210.xml">
```

x

Les Espaces de Noms XML

- **Éviter les conflits de noms d'élément et d'attribut**
  - Ré-utilisation de l'information (document, DTD et schéma)
  - Composition de fragments XML (bien-formés)
  - Principe : mettre un drapeau pour indiquer sur quel terrain on se trouve
- **Déclaration :**

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
   <html>
        <head><title>exemple</title></head>
        <body>
             <h1>exemple</h1>
        </body>
   </html>
  </xsl:template>
</xsl:stylesheet>
```

x

Espaces de Noms Réservés

- **Le préfixe `xml:` est réservé pour certains attributs :**

```
<title xml:space="default">...</title>

<p xml:lang="FR">...</p>
```

- **Forme de l'attribut xml:space**

```
<!ATTLIST nomElement

          xml:space (default | preserve) 'preserve'>
```

- **On ne le déclare pas … théoriquement**

Les Schémas XML

- **Recommandation : 02 May 2001**
  - **Primer : www.w3.org/TR/xmlschema-0**
  - **Structures : www.w3.org/TR/xmlschema-1**
  - **Types : www.w3.org/TR/xmlschema-2**
- **Motivations :**
  - Remplacer les DTDs par un mécanisme plus adapté à XML
- **Particularités :**
  - Un schéma est lui-même un document XML
  - Types de base plus riches
  - Espaces nominaux pris en charge (pas le cas dans les DTD)
- **Schémas RelaxNG (*relaxing*): le bébé de James Clark !**
  - Syntaxe XML et syntaxe simplifiée
  - Proches de l'esprit des DTD + types simples
  - Pas de typage complexe
  - Future norme ISO/IEC 19757-2

Schémas: exemples

```
<!ELEMENT Address (name, street, city, zip)>
<!ATTLIST Address country CDATA #IMPLIED>
```

```
<xsd:complexType name="Address">
        <xsd:sequence>
                <xsd:element name="na
type="xsd:string"/>
                <xsd:element name="st
type="xsd:string"/>
                <xsd:element name="ci
type="xsd:string"/>
                <xsd:element name="zi
type="xsd:decimal"/>
        </xsd:sequence>
        <xsd:attribute
                name="country"
                type="xsd:NMTOKEN"/>
</xsd:complexType>
```

```
<element name= "Address">
        <element name="name">
                <text/>
        </element>
        <element name="street">
                        <text/>
        </element>
        <element name="city">
                <text/>
        </element>
        <element name="zip">
                        <text/>
        </element>
        <attribute name="country">
                <text/>
        </attribute>
 </element>
```

x

Formations lucid'i.t.

Survol des techniques

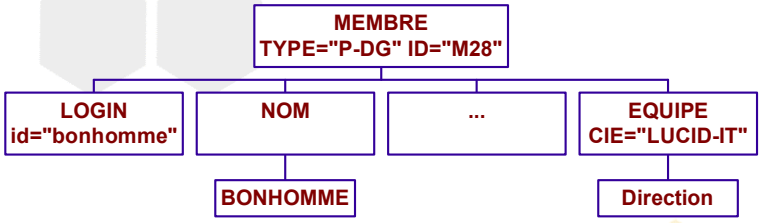
Chemins, liens et pointers XML

XPath

- **Langage non XML**
  - Xpath 1.0 : Recommandation du W3C du 16 novembre 1999
- **Définir une syntaxe pour accéder aux éléments d'un document XML**
  - Utilisable avec XSL-T et XPointer
  - Document représenté sous forme arborescente
- **2 notations :**
  - Complète
    - /child::voiture/attribute::marque
  - Abrégée
    - /voiture/@marque

x

Une structure XML représente un arbre hiérarchique



x

XPath - Exemples

```
<DB>
  <MEMBRE TYPE = "P-DG" ID = "M28">
    <LOGIN ID = "bonhomme"/>
    ...
    <EQUIPE CIE = "LUCID-IT"> Direction </EQUIPE>
  </MEMBRE>
  <MEMBRE TYPE = "ING" ID = "M14">
    <LOGIN ID = "julien"/>
    ...
  </MEMBRE>
</DB>
```

/ ou /DB /DB/MEMBRE /DB/MEMBRE[2]

/DB/MEMBRE[@ID='M28']/EQUIPE[1]/text()

/DB/MEMBRE/LOGIN[@ID='julien']/../@ID

x

XLink

- **Recommandation du W3C (27 juin 2001)**
- **XLink : définir des liens entre documents XML**
- **Liens simples : 1 source – 1 cible**
- **Liens complexes**
  - Plusieurs cibles
- **Syntaxe XML**
  - Ensemble d'attributs pré-définis

```
<lien xlink:type="simple"
      xlink:href="http://www.w3.org/">W3C</lien>
```

x

XML-Base

- **Recimmandation du W3C (27 juin 2001) v1.0**
- **But : Permettre de définir une adresse de base pour tous les liens sous-éléments**
  - Comme la balise **<base>** en HTML
- **Principe : ajout d'un attribut à un élément**

```
<monElement xml:base="URI de base"/>
```

x

XPointer - Adressage XML

- **Proposé à la recommandation du W3C (13 November 2002)**
- **Sert à désigner un élément à l'intérieur d'un document**
  - comme HTML, il y a des ancres :

```
<A NAME="TOTO">http://www.titi.fr/index.html#TOTO</A>
```

- **Dans XML, les pointeurs peuvent être des IDs comme "M28" ou des expressions comme :**

```
http://…/doc.xml#xptr(id(M28))
http://…/doc.xml#xptr(/DB/MEMBRE[28]/MEL)
```

```
<link href="document.xml#xpointer(expression Xpath)"/>
```

x

Formations lucid'i.t.

Transformation et style

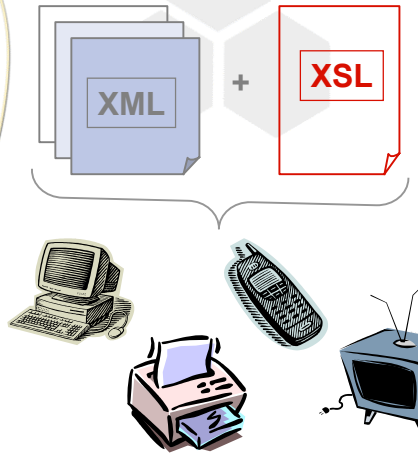
XSL/XSL-T et XSL-FO

global management system

Afficher des documents XML

- **Un document XML ne fournit pas d'information sur sa présentation**
- **Affichage personnalisé**
- ➜ **Feuilles de style :**
  - *Casdading Style Sheets* (CSS 1 et 2)
  - *Extensible Style Language* (XSL)
- **Transformation de documents XML**

eXtensible Style Language

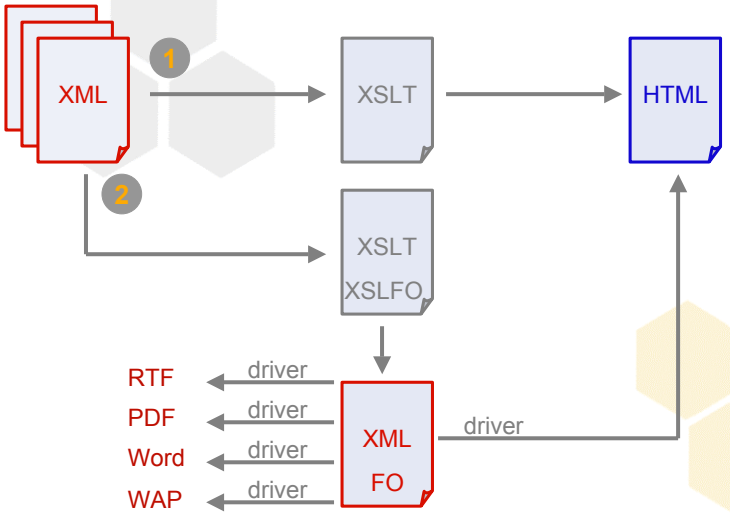


- **Décrit la manière dont les documents XML seront affichés, imprimés ou … prononcés**
- **Indépendant du média de sortie**

Aspects fondamentaux de XSL

- **XSL = Transformation + Propriétés d'Affichage**
- **XSL-T : Transformation de documents XML**
  - Transformer un doc XML en un autre doc XML
  - Par défaut : production de documents HTML (BF !)
- **XSL-FO : Formatage des données/objets XML**
  - Les Formatting Objects
  - Indépendant (Word/RTF, PS, PDF, MIF, …)

Architecture traitements XSL



Une feuille de style...

- **Cher moteur XSL, quand tu rencontres la racine du document, fait quelque chose...**
- **Cher moteur XSL, si tu rencontres l'élément `annuaire`, fait telle chose...**
- **Cher moteur XSL, si tu rencontres l'élément `membre`, fait telle autre chose...**
- **Et ainsi de suite ...**

➔ Une feuille de style XSL est une suite de règles (*templates*).

Exemple de structure XSL

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  ...
  <xsl:template match="/">
    ...
  </xsl:template>

  <xsl:template match="annuaire">
    ...
  </xsl:template>
  ...
</xsl:stylesheet>
```


Structure des règles (template)

- **Chaque règle possède deux parties**
  - Un motif (XPath) qui identifie le/les nœud(s) XML du document qui est/sont concerné(s) par la règle et sur le(s) quel(s) il faut appliquer une action
  - Une action qui effectue la transformation et/ou spécifie les caractéristiques de la présentation

```
<xsl:template match= 'un motif'>
  [action]
</xsl:template>
```

Structure d 'une règle (exemple)

```
<xsl:template match='annuaire'>
  <xsl:apply-templates/>
</xsl:template>
```

Cher moteur XSL, quand tu *parses* un document XML et que tu tombes sur un élément **<annuaire>** , utilise cette règle

Va vers chacun de mes fils (**<entete>** et **<membre>**) et applique les règles qui les concernent

Et encore bien d'autres choses

- **Nous n'avons pas parlé de :**
  - SAX (Simple API for XML)
  - DOM (Document Object Model)
  - SVG : Image Vectorielle en XML
  - XQuery : langage de requête pour documents XML
  - XForms : Formulaires en XML
  - XHTML : Adaptation de HTML à XML
  - VoiceXML
  - XML-RPC et SOAP
  - UDDI : Registre et directory en XML
  - etc.