4. XML

1. Introduction

Dans cette section, nous allons nous intéresser à la technologie XML (*Extensible Markup Language)*. En exposant certains de ses concepts et notions. Nous allons commencer par la présenter puis relater un peu de ses origines puis la confronter à ses différents homologues tels que HTML et SGML. Nous aborderons ensuite les principes de base de XML puis sa structure.

2. Présentation de XML

XML, conçu initialement comme un langage de description de données, destiné à la publication électronique sur le Web, est amené à jouer un rôle fondamental dans l'architecture des applications informatiques s'appuyant sur l'architecture Web. XML est de plus en plus souvent utilisé pour représenter des données (structurées et non structurées) et y accéder, et plus encore, pour décrire et exécuter à distance des traitements ou des opérations.

1.1 Définition de XML

X : *comme eXtensible*

- Possibilité de définir les balises.
- Balises pour structurer.

M : *comme Markup*

- Tous les éléments sont repérés au moyen de balises.
- Les balises servent à structurer le document.

L : *comme Language*

- Langage de description de documents.
- Règles à respecter.

1.2 Différences entre HTML et XML

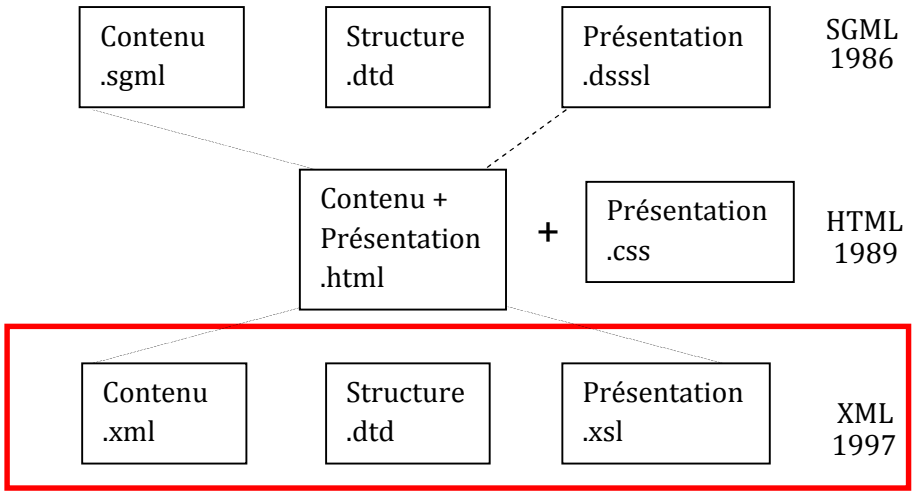
- HTML est un langage très bien adapté à la **diffusion d'informations** sur le réseau.
- Nouveaux besoins pour le Web :
  - ✓ Commerce en ligne
  - ✓ Adaptation du contenu au support (téléphones mobiles, assistants personnels, ...)
  - ✓ Intégration du Web dans les systèmes d'information des entreprises.
  - ✓ Echange de données : serveurs Web <-> autres applications (SGBD, tableurs, ...)
  - ✓ ...

- XML (comme HTML) est un langage de balisage (*markup*), c'est-à-dire un langage qui présente de l'information encadrée par des balises.
- Mais contrairement à HTML, qui présente un jeu limité de balises orientées présentation (titre, paragraphe, image, lien hypertexte, etc.).
- XML est un ***métalangage,*** qui va permettre d'inventer à volonté de nouvelles balises pour isoler toutes les informations élémentaires (titre d'ouvrage, prix d'article, numéro de sécurité sociale, référence de pièce…) ou agrégats d'informations élémentaires, que peut contenir une page Web.

```
<restaurant>
    <nom> la rose </nom>
    <telephone>  031 45 46 47 </telephone>
    <adresse>  Rue de Benmhidi </adresse>
</restaurant>
```

1.3 Structure d'un document XML

Dans le principe de base d'un document XML, il faudra s'axer sur les points suivant pour construire un document XML valide, souple et bien conforme.



DTD

Document Type Définition est un fichier écrit en XML qui contient une définition formelle d'un type particulier de documents; elle définit les noms qui peuvent être utilisés pour les types d'éléments, où ils peuvent apparaître et comment ils s'arrangent les uns par rapport aux autres.

Un document XML peut être valide ou conforme à sa classe selon qu'il comporte ou non une DTD.

XSL

Le XSL (Extensible Style Language) est le langage utilisé pour définir les feuilles de style qui seront associées aux documents XML. C'est le fichier XSL qui permettra de définir que tel élément XML doit être affiché avec telle fonte, de telle couleur, etc. Ces décisions seront, grâce à XSL, prises par le créateur du document qui aura ainsi un meilleur contrôle sur l'apparence de son document. Il pourra également faire référence à un fichier XSL public déjà existant (réutilisabilité). Le XSL s'inspire de 2 normes de feuilles de style, le CSS (Cascading Style Sheet), qui est utilisé avec des fichiers HTML et le DSSSL (Document Style Semantics and Specification Language) qui est une norme de feuilles de style moins complexe.

En résumé, les documents suivants sont requis :

- Définition de type de document (DTD), l'élément référence en XML, extension .dtd ;
- Le document .xml proprement dit, extension .xml ;
- Un document de mise en page, si nécessaire, extension .xsl.

Le minimum requis c'est le document XML lui-même.

Si vous avez rempli les deux premières conditions, c à d si vous avez défini les balises dans la DTD et les avez utilisées telles quelle en XML, on dit que votre document est « valide ». Il suffit que vous enfreignez une seule des strictes règles régissant XML pour que le programme de visualisation interrompe le chargement et affiche un message d'erreur.

Règles strictes en XML

- Respecter la casse des balises <individu> ≠ <INDIVIDU>.
- Eviter les erreurs d'imbrication <a><w> </w>.
- A chaque balise de début correspond une balise de fin.
- Toutes les balises utilisées sont définies dans la DTD
- Elles sont utilisées dans l'ordre spécifié dans la DTD

Création d'un document XML

Un document est composé

- D'un prologue (facultatif)
- D'un arbre d'éléments (obligatoire : décrit le con tenu)
- De commentaires et instructions de traitement (facultatif)

Exemple

```
<?xml version="1.0" encoding='ISO-8859-1' standalone='yes'>
<personne>
   <nom>Martin</nom>
   <prenom>Jean</prenom>
   <adresse>
      <rue>rue du Bac</rue>
      <ville>Paris</ville>
   </adresse>
<!-- pas d'autre information disponible -->
</personne>
```

a) ***Prologue*** : déclaration XML

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

xml version="1.0": décrit la version utilisée
encoding='ISO-8859-1' : codage de caractères utilisés dans le document
standalone='yes': existence de déclarations extérieures
(yes= toutes les déclarations nécessaires au document sont incluses)

b) ***Arbre d'éléments*** <nom>contenu</nom>

- Un document est formé d'une hiérarchie d'éléments dénotant la structure sémantique du contenu.
- Tout élément fils est complètement inclus dans son père.
- L'élément racine est unique et contient tous les autres éléments.
- Un élément peut contenir d'autres éléments, des données, …etc. Il peut aussi être vide.
- Le nom de l'élément peut éventuellement être suivi d'attributs qui décrivent des propriétés de l'élément. Chaque attribut a une valeur unique.

Un document XML est bien formé s'il respecte ces principes.

2. DTD

Une DTD peut être soit un fichier externe, soit incluse dans le document XML. Nous avons deux types de définition : externe et interne. Dans la DTD, nous allons éventuellement spécifier des éléments, des attributs, des entités et des notations. Elle peut aussi contenir des commentaires.

S'il s'agit d'un fichier externe, elle est spécifiée dans le document XML à l'aide d'une Déclaration de Type de Document suivant la syntaxe dans l'exemple :

```
<! DOCTYPE nom statut URL>
```

Rôle des DTD

- Modèle selon une organisation hiérarchique (définition deséléments, attributs, contenus).
- Spécifie la structure des instances de documents : cet élément contient ces éléments, ces attributs, etc.
- Spécifie le type de données de chaque élément et attribut.
- Définition d'entités : mécanisme d'inclusion, utile pour les opérations de modularisation et de réutilisation (Un document peut être découpé en **entités** enregistrés dans un ou plusieurs fichiers).

Noms de balise

- Les noms de balise sont libres et peuvent comprendre :
  - Des lettres de l'alphabet (y compris les lettres accentuées)
  - Des chiffres
  - Les caractères – et _
- Ils ne doivent pas contenir d'espace, et ne peuvent pas commencer par un chiffre.
- Les majuscules sont distinguées des minuscules.

2.1 Eléments

Un élément est composé :

- D'une **balise de début** qui contient le nom de l'élément et éventuellement ses attributs,
- D'un contenu
- D'une balise de fin

Par exemple

<rapport langue="français">présentation d'XML </rapport>

- Balise de début : **<rapport langue="française">**
- Nom : **rapport**
- Attribut : **langue="française"**
- Contenu : présentation **d'XML**
- Balise de fin : **</rapport>**

Déclaration d'élément

- Définition d'un élément dans une DTD :
  <!ELEMENT nom modeledecontenu>

- Il existe plusieurs types de déclarations de type d'éléments. Nous allons essayer de les voir dans le tableau ci-dessous :

TYPE D'ELEMENT	*DECLARATION*
Déclaration de l'élément vide.	< ! ELEMENT nom EMPTY > EMPTY signifie que l'élément est vide
Déclaration de l'élément contenant des données textuelles.	< ! ELEMENT nom (#PCDATA)> #PCDATA signifie données textuelles
Déclaration de l'élément libre.	< ! ELEMENT nom ANY > ANY signifie que l'élément contient tous ce que l'on veut
Déclaration de séquencement d'élément.	< ! ELEMENT nom (prem1,prem2,…)> séquence d'élément fils pour l'élém père
Déclaration de choix entre élément.	< ! ELEMENT nom (choix3| choix7|rouge)
Déclaration d'élément au contenu Ambigu entre choix et séquence.	< ! ELEMENT nom (choix1,choix8) | (choix1, choix3)>

- Modèle de contenu vide **EMPTY** :
  - Obligatoirement vide (infos uniquement dans attributs)
  - Ne peut pas être composé (pas d'éléments fils)
  - Ex : **<!ELEMENT note EMPTY >**

- Texte simple : **#PCDATA** *Parsed Character Data* (pas d'éléments fils)
  Ex: **<!ELEMENT titre (#PCDATA)>**
- Composé d'éléments:
  - Décrit l'organisation des sous-éléments directs (ie. les fils)
  - Deux organisations possibles :
    - La séquence (fils1**,** fils2, …) : fils1 **est suivi de** fils2 qui est suivi de ...
    - Le choix (fils1**|** fils2 …) :fils1 **ou** fils2 ou ...
- On peut décrire l'organisation par morceaux: blocs entre parenthèses
- Ex : **<!ELEMENT chapitre (titre, intro, (texte-long | résumé))>**

Indicateur d'occurrence

Il existe trois types d'indicateurs d'occurrence d'élément qui sont comme suit :

- *L'indicateur(*) :* indicateur qui indique que l'élément peut être omis ou qu'il peut apparaître plusieurs fois.
- *L'indicateur(+) :* indicateur d'élément qui montre que l'élément apparaît au moins une fois ou qu'il peut le faire plusieurs fois.
- *L'indicateur(?) :* indicateur qui indique que l'élément peut être omis ou qu'il n'apparaît qu'une seule fois.
- rien : exactement une fois

Exemple

```
<!ELEMENT liste_livres (livre+)>
<!ELEMENT livre
(titre,auteur+,editeur,description?,prix)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT editeur (#PCDATA)>
<!ELEMENT description (#PCDATA)>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE liste_livres SYSTEM "livres.dtd">
<liste_livres>
<livre>
<titre>Votre site Web avec Office 2000</titre>
<auteur>Herbert Bauer</auteur>
<editeur>MICRO APPLICATION</editeur>
<prix>132,00</prix>
</livre>
<livre>
<titre>Grand Livre Office 2000</titre>
<auteur>Helmut Kraus</auteur>
<auteur>Helmut Vonhoegen</auteur>
<editeur>MICRO APPLICATION</editeur>
<description>La suite bureautique de Microsoft dans ses moindres
détails</description>
<prix>167,00</prix>
</livre>
</liste_livres>
```

2.2 Attributs d'un élément

- Les attributs sont un **autre** moyen de représenter l'information
- L'ordre des attributs n'est pas important
- Un attribut doit toujours avoir une valeur, encadrée par des apostrophes (simple ou double).
- Il ne peut pas y avoir deux attributs de même nom dans un élément.

- Déclaration dans la DTD :
  <!ATTLIST nom-élément nom-attribut type-attribut mode-défault>
- On peut grouper la déclaration des attributs d'un même éléme
- **Type-attribut** :
  - CDATA : la valeur de l'attribut est une chaîne de caractères,
  - ID : identificateur d'élément,
  - IDREF : renvoi vers un ID existant,
  - IDREFS : renvoi vers un ensemble d'ID existants,
  - NMTOKEN(S) : un ou des noms symboliques (sans blanc),
  - (a | b | c…) : type énuméré.
- **Mode** :
  - Valeur par défaut
  - Obligatoire : #REQUIRED
  - Facultatif : #IMPLIED
  - Constante : #FIXED

Exemple d'attributs :

- Dans la DTD :
  <!ELEMENT date (#PCDATA)>
  <!ATTLIST date format (ANSI | ISO| FR) #REQUIRED>
- Dans le document
  <date format='FR'>7 mars 2008 </date>
  <date format='ISO'>2008-3-7 </date>

Exemple

DTD

```
<!ELEMENT bibliothèque  (personne+, livre+)>
<!ELEMENT personne  (nom)>
<!ATTLIST personne num ID #REQUIRED>
<!ELEMENT livre  (titre, auteur+)>
<!ELEMENT auteur EMPTY>
<!ATTLIST auteur ref IDREF #REQUIRED>
<!ELEMENT nom (#PCDATA) >
<!ELEMENT titre (#PCDATA) >
```

XML

```
<bibliothèque>
     <personne num="p1"><nom>Toto</nom></personne>
     <personne num="p2"><nom>Titi</nom></personne>
     <livre>
      <titre>XML en 2 jours </titre>
      <auteur ref="p1"/>
      <auteur ref="p2"/>
     </livre>
</bibliothèque>
```

Exemple d'attributs de type IDREFS

DTD

```
<!ELEMENT bibliothèque  (personne+, livre+)>
<!ELEMENT personne  (nom)>
<!ATTLIST personne num ID #REQUIRED>
<!ELEMENT livre  (titre, auteurs)>
<!ELEMENT auteurs EMPTY>
<!ATTLIST auteurs ref IDREFS #REQUIRED>
<!ELEMENT nom (#PCDATA) >
```

Document XML

```
<bibliothèque>
     <personne num="p1"><nom>Toto</nom></personne>
     <personne num="p2"><nom>Titi</nom></personne>
     <livre>
      <titre>XML en 2 jours </titre>
      <auteurs ref="p1 p2"/>          liste de valeurs
      existantes séparées par un espace
     </livre>
</bibliothèque>
```

Un document est valide s'il est conforme à une DTD

Limites des DTD :

- Pas de possibilités de typer les contenus
- Typage faible des valeurs d'attributs
- Les DTD ne sont pas suffisantes pour l'échange de données structurées (BD, commerce électronique, etc.).

XML schéma

2.3 XML schema

Un schéma doit permettre de spécifier le même genre d'informations qu'une DTD, c'est-à-dire la syntaxe d'une classe de documents XML, en définissant les éléments et attributs ainsi que les contraintes sur les valeurs de ceux-ci.

Définition de l'élément

Chaque élément déclaré dans un schéma à un type associé qui définit le genre de contenu que l'élément en question peut avoir:

```
<xsd:element name="LeNomDeMonElement"type="LeTypeContenuDel'Element">
```

Différences entre DTD et SCHEMA

Le langage avec lequel la DTD est écrite possède des inconvénients : syntaxe différente de XML, pas de notions d'espace de noms (d'où risque de collisions entre symboles importés et symboles définis localement), pour cela l'XML Schéma propose :

- La syntaxe de l'XML Schéma est écrite en langage XML, c'est une des plus évidentes différences entre Schéma et DTD.
- Typage de données : l'XML Schéma propose plus de type de données que ceux proposées dans la DTD ; de plus que la DTD l'XML Schéma propose des Types Complexes, et des types Objet.
- L'extensibilité n'est pas offerte par la DTD est quand on pense que le langage XML signifie eXtensible, et que l'extensibilité ne l'est pas a cent pour cent. L'XML Schéma offre cette extensibilité à cause des types de données dérivées par l'utilisateur et la possibilité de définir des structures complexes fournissant des moyens évolutifs d'étendre des schémas.

2.4 Liens et chemins (XLL)

Le XLL (Extensible Link Language) est le langage de description des liens hypertextes en XML. XLL constitue la deuxième partie de la norme XML. XLL est basé sur la norme ISO 10744, (Technologies de l'information: Langage de structuration hypermédia/événementiel).

En XLL est composé de deux parties qui sont XlinK et Xpointer.

a) <u>XLinK</u>

XLink est un mécanisme de liaison similaire aux liens hypertexte HTML, défini pour relier des documents XML. Contrairement aux liens hypertexte HTML, il permet de créer des liens bidirectionnels ou de lier plusieurs documents à l'aide d'un même lien.

b) <u>XPointer</u>

XPointer est un mécanisme utilisé pour pointer vers des sous-ensembles de documents XML. Il fonctionne de façon similaire aux ancres HTML, mais XPointer offre une

bien plus grande souplesse, permettant de sélectionner des nœuds ou des ensembles de nœuds dans le document cible à l'aide de XPath.

Exemple XLink

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<lien>
<renvoi xml:link="simple" href="http://www.yahoo.com">Yahoo</renvoi>
<renvoi xml:link="simple" href="bonjour.xml">Bonjour</renvoi>
</lien>
```

2.5 Parseurs (processeur ou analyseurs)

L'analyseur généralement appelé ***parseur*** est un outil logiciel permettant de parcourir un document et d'en extraire des informations.
Dans le cas de XML (on parle alors de *parseur XML*), l'analyseur permet de créer une structure hiérarchique contenant les données contenues dans le document XML.

On distingue deux types de parseurs XML :

- les parseurs validants (validating), permettant de vérifier qu'un document XML est conforme à sa DTD
- les parseurs non validants (non-validating) se contentant de vérifier que le document XML est bien formé (c'est-à-dire respectant la syntaxe XML de base)

Les analyseurs XML sont également divisés selon l'approche qu'ils utilisent pour traiter le document. On distingue actuellement deux types d'approches :

- Les API utilisant une approche **hiérarchique** : les analyseurs utilisant cette technique construisent une structure hiérarchique contenant des objets représentant les éléments du document, et dont les méthodes permettent d'accéder aux propriétés. La principale API utilisant cette approche est **DOM** (*Document Object Model*)
- Les API basés sur un mode **événementiel** permettent de réagir à des événements (comme le début d'un élément, la fin d'un élément) et de renvoyer le résultat à l'application utilisant cette API. **SAX** (Simple API for XML) est la principale interface utilisant l'aspect événementiel.

DOM

DOM (*Document Object Model*) est une spécification du W3C (*World Wide Web Consortium*) définissant la structure d'un document sous forme d'une hiérarchie d'objets, afin de simplifier l'accès aux éléments constitutifs du document.

SAX

SAX est une API basée sur un modèle événementiel, cela signifie que SAX permet de déclencher des événements au cours de l'analyse du document XML.
Une application utilisant SAX implémente généralement des gestionnaires d'événements, lui permettant d'effectuer des opérations selon le type d'éléments rencontrés.

2.6 Sémantique (RDF, RDFS)

RDF (Resource Description Framework) est un modèle, associé à une syntaxe, dont le but est de permettre à une communauté d'utilisateurs de partager les mêmes métas donnés pour des ressources partagées. Il a été conçu initialement par le W3C pour permettre de structurer l'information accessible sur le web et de l'indexer efficacement.

RDF n'est pas particulièrement conçu pour permettre de stocker les métas données de documents mais plutôt pour permettre leur échange et leur traitement par des opérateurs humains ou artificiels. Un des gros avantages de **RDF** est son extensibilité, à travers l'utilisation des schémas **RDF** qui peuvent s'intégrer et ne s'excluent pas mutuellement grâce à l'utilisation du concept d'espace de nom (« namespace »).

La construction de base en **RDF** est le triplet {propriété, ressource, valeur} spécifiées généralement en **XML**. De même, les propriétés peuvent être des éléments **XML** ou bien des attributs **XML**.

Tout ce qui est exprimé avec **RDF** est appelé une ressource. La ressource est toute entité d'information pouvant être référencée en un bloc, par un nom symbolique (littéral) ou un identificateur. L'identificateur est obligatoirement un **URI**.

Une propriété est un aspect spécifique, une caractéristique, un attribut ou une relation utilisée pour décrire une ressource. Chaque propriété a une signification spécifique, des valeurs autorisées, des types de ressources qu'elle peut décrire et des relations avec d'autres propriétés.

Une ressource spécifique associée à une propriété et sa valeur correspondante est une expression ou déclaration (« statement ») **RDF**. Ces trois parties (ressource, propriété, valeur) sont appelées respectivement sujet, prédicat et objet.