**Objectif du cours...**

* Connaître les principes de base du XML.
* Savoir rédiger un fichier XML, ainsi que la DTD associées.
* Savoir reconnaitre un document bien formé, valide.

|  |  |
| --- | --- |
| **1** | Petite présentation de XML, comparaison entre XML et HTML. |

Pour commencer, voici un exemple de **document XML** :

|  |  |
| --- | --- |
| 1  2  3  4  5 | <?xml version="1.0" ?>  <informations>    <auteur type="salarie">Bob</auteur>    <texte>Une nouvelle information</texte>  </informations> |

On remarque à travers cet exemple que :

* XML est un langage à base de **balises**, comme le HTML (les **éléments**, comme "informations", sont notés sous forme de balise ou marqueurs). Ici on peut voir les éléments **informations**, **auteur**, **texte**. Les mots suivants sont quasiment synonymes : **balises**, **éléments**, **marqueurs**.
* De la même façon que le HTML, il comporte, en plus des éléments :
  + des **attributs** (par exemple **type**)
  + des **valeurs** pour ces attributs (par exemple **salarie**).
* Il y a des guillemets autour des valeurs des attributs (par exemple **"salarie"**). C'est obligatoire, contrairement au HTML. On verra plus loin qu'en règle générale, le XML est beaucoup plus strict que le HTML au niveau de la syntaxe.
* Les entités (**informations**, **auteur**, **texte**) ne sont pas prédéfinies dans le langage XML mais ont été librement inventées pour cet exemple. C'est une particularité de XML qui le rend si différent de HTML : on peut créer nos propres balises, et donc créer nos propres langages ! En HTML (4), on se contente d'utiliser les 95 balises prédéfinies : **<html>**, **<head>**, **<body>**, **<table>**, etc.

XML est donc globalement relativement différent du HTML, il n'en est d'ailleurs absolument pas le successeur. Voici les **différences** :

|  |  |
| --- | --- |
| **XML** | **HTML** |
| * Ils sont tous les deux issus du **SGML**. * Ils s'écrivent avec des **balises**. * Ce sont des formats en **texte brut** (et non pas binaires), on peut donc les consulter dans des éditeurs de texte brut (par exemple : **Notepad**). * Ils sont **indépendants d'une plateforme**. | |
| Le **XML** sert à **décrire des données quelconques**, de façon **structurée**. | Le **HTML** sert à **décrire des pages Web** destinées aux navigateurs. |
| Le **XML** est un **générateur de langages**(un métalangage). | Le **HTML** est un **langage statique**, figé. |
| Le **XML** est un langage **strict** donc l'écriture doit être rigoureuse. | Le **HTML** est devenu **très permissif** à cause des navigateurs récents. |
| Le **XML** se veut **ouvert à toutes les plateformes**, notamment les mobiles, les pockets, etc. | Le **HTML** est essentiellement destiné aux **microordinateurs**. |
| Le **XML** est un langage promis à un bel avenir. | Le **HTML** est en bout de course. |

|  |  |
| --- | --- |
| **2** | Présentation détaillée de XML. |

**XML** signifie e**X**tensible **M**arkup **L**anguage. La [**norme XML**](http://www.w3.org/XML/) est définie par le **W3C** (**W**orld **W**ide **W**eb **C**onsortium, [**www.w3.org**](http://www.w3.org/)), depuis **1996**.

Comme le HTML, **XML** est dérivé du langage SGML (**S**tandard **G**eneralized **M**arkup **L**anguage, norme [**ISO 8879**](http://www.iso.ch/cate/d16387.html)). Le SGML date des balbutiements d'Internet (années 80), c'est un langage concis et abstrait, difficile à utiliser tel quel. Le XML représente une version simplifiée, abrégée du SGML, donc plus abordable. Le XML doit donc plutôt être considéré comme une simplification du SGML, plutôt qu'une amélioration du HTML.

Le **XML**, en tant que **métalangage**, permet de créer de nouveaux langages. On peut soi-même créer notre propre langage pour nos applications personnelles. Voici des exemples de langages créés à partir de XML et universellement utilisée, des exemples propriétaires et personnels :

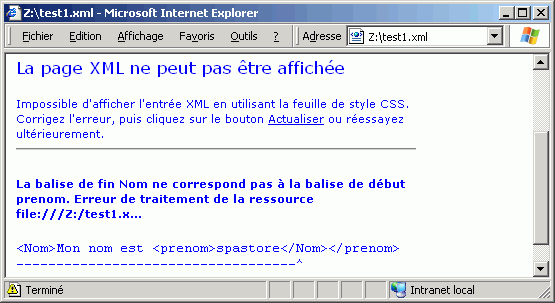
* **XHTML** (e**X**tended **H**yper**T**ext **M**arkup **L**anguage, pour décrire les pages Web. C'est lui le vrai **successeur du HTML**, il correspond en effet à une reformulation du HTML 4.0, mais en respectant la syntaxe et les règles du XML.
* **MathML**, pour représenter les formules mathématiques, [**exemple sur Wikipedia**](http://fr.wikipedia.org/wiki/MathML). Malheureusement, actuellement, trop peu de navigateurs savent afficher correctement ce format (par exemple : le navigateur **Amaya**).
* **XMLTV**, pour représenter les programmes de télévision.



 XML, en soi, ne permet pas de faire grand-chose ! Il permet en fait uniquement de décrire des données sous une forme bien structurée, mais pas la façon dont elles sont représentées. Pour dire comment un document XML doit s'afficher dans un navigateur, on a recourt à d'autres technologies (**CSS**, **XSL** ou**XSLT**).

|  |  |
| --- | --- |
| **3** | La syntaxe du XML, notion de document "bien formé". |

Comme on l'a déjà dit, le XML obéit à une **syntaxe très stricte** (beaucoup plus que celle du HTML). La moindre erreur entraînera l'affichage, dans le navigateur, d'une erreur de syntaxe, en lieu et place du document.



Un tel document est dit **mal formé**. Un document qui obéit aux règles syntaxiques est dit **bien formé**.

Voici les **règles de syntaxe à respecter** pour qu'un document soit **bien formé** :

* il contient une en-tête du type **<?xml version="1.0" ?>** ;
* il existe un seul élément racine qui contient tous les autres éléments, par exemple :
  + en **XHTML**, on a la balise racine **<html>** :  **<?xml version="1.0" ?>  
        <html>  
         ...  
         </html>**
  + en **XMLTV**, on a la balise racine **<tv>** :  **<?xml version="1.0" ?>  
        <tv>  
        ...  
        </tv>**
* la casse est respectée sur les balises. Par exemple :
  + **<table> ... </TABLE>** n'est pas bien formé en XML (alors que c'est accepté en HTML) ;
  + **<table> ... </table>** est bien formé ;  
    *Remarque : actuellement, la tendance est d'écrire les balises systématiquement****en minuscule****pour éviter les erreurs. Ce n'est pas une obligation.*
* les balises sont correctement imbriquées et ne se chevauchent pas. Par exemple :
  + **texte en <b>gras, <i>gras italique</b>, italique</i>.**, n'est pas bien formé (cf. erreur ci-dessus), alors que cela s'affiche bien en HTML (texte en **gras, *gras italique****, italique*.) ;
  + L'équivalent bien formé de la structure ci-dessus est : **texte en <b>gras, </b><b><i>gras italique</i></b><i>, italique</i>.** ;
* A chaque balise ouvrante correspond une balise fermante, par exemple :
  + **<table> ... </table>**, **<img ...></img>** est bien formé ;
  + **<img ... />**, **<meta ... />** sont bien formés ;
* La valeur des attributs est entre guillemets. Par exemple :
  + **<table width=150>** n'est pas bien formé en XML (alors que c'est accepté en HTML).
  + **<table width="150">** est bien formé.
* Les valeurs des attributs sont obligatoires. Par exemple :
  + **<table border>** n'est pas bien formé en XML (alors que c'est accepté en HTML).
  + **<table border="1">** est bien formé.
* Les noms (des éléments, des attributs) doivent respecter les règles suivantes, proches du Javascript :
  + Les noms peuvent contenir des lettres (y compris les accents et les alphabets spéciaux), des chiffres, point, deux points, tiret, souligné.
  + Les noms doivent commencer par une lettre ou un souligné.
  + Les noms ne peuvent commencer par les lettres xml (ou XML ou Xml...).
  + Les noms ne peuvent contenir des espaces.
  + La longueur des noms est libre mais on conseille de rester raisonnable.
  + On évitera certains signes qui pourraient selon les logiciels, prêter à confusion (**- ; . < >** etc.)
  + Les caractères accentués (é, à, ê, ï, ù, etc.) sont autorisés mais pourraient être mal interprétés par certains programmes.

**Rappel sur une erreur fréquente...**

Les balises XML (comme en HTML) ne peuvent pas contenir d'espace. Par exemple, si je vois :

**<table border>**

Je sais qu'il s'agit de la balise **table** et de l'attribut **border**, et non pas de la balise **~~table  border~~**, ce qui est syntaxiquement impossible.

Soit le document XML suivant :

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

**<informations>**

**<auteur type=salarie>Bob</auteur>**

**<texte>Une nouvelle information</texte>**

**</information>**

Est-ce qu'il est bien formé (si non, donnez la liste des erreurs) ?

Réécrivez le document bien formé :

Dites combien vous voyez d'erreurs qui font que ce document est mal formé, relevez et expliquez chacune des erreurs :

**<?xml version="1.0" encoding="iso-8859-1" ?>**

**<client>**

**<nom>Bob</nom>**

**<prénom>Billy</prénom>**

**<adresse>20 allée des lilas, 84000 Avignon</adresse>**

**<date-adhesion>25/02/2004</date-adhesion>**

**</client>**

**<client>**

**<nom>Bugard</nom>**

**<prénom>Frank</prénom>**

**<adresse>12(bis) rue des romarins, 84000 Avignon</Adresse>**

**<date-adhesion>14/12/2005</date-adhesion>**

**</client>**

**<client type=professionnel>**

**<nom>Rorjak</nom>**

**<prénom>Marcel</prénom>**

**<adresse-postale>5 route des "Francs-tireurs", 13001 Marseille</adresse-postale>**

**<date-adhesion>19/04/2007</date-adhesion>**

**</client>**

**<client>**

**<nom>Dubois</nom>**

**<prénom>Albert</prénom>**

**<adresse>albert.dubois@free.fr</adresse>**

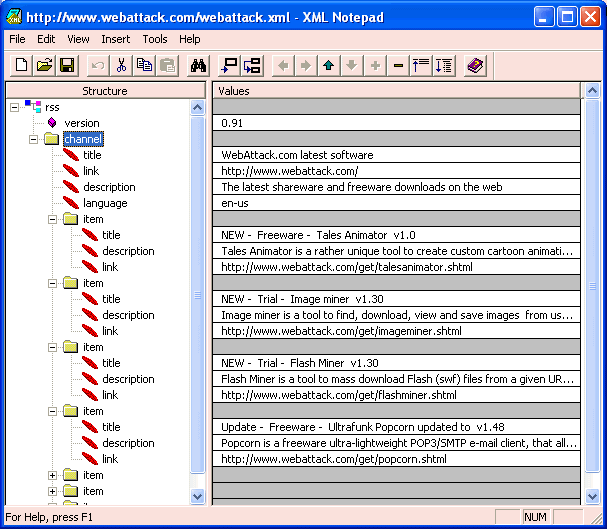
**<telephone portable="06 11 23 95 45" fixe="09 62 45 78 51">**

**<date-adhesion></date-adhesion>**

**<\_facture\_>...</\_facture\_>**

**</client>**

|  |  |
| --- | --- |
| **4** | Les éditeurs XML. |

* Ils permettent, selon les logiciels :
  + de créer des documents XML et tous les autres documents associés (DTD, XSL, etc.) ;
  + de vérifier si les documents XML sont bien formés et valides ;
  + de visualiser le résultat de la transformation XSL ;
  + de visualiser l'arbre correspondant à notre fichier XML ;
  + etc.
* Exemple : **XML NotePad** de **Microsoft** ([**http://www.snapfiles.com/get/xmlnotepad.html**](http://www.snapfiles.com/get/xmlnotepad.html)), freeware, plutôt pauvre en fonctionnalités :  
    
  **[](http://cours.sebastien-pastore.com/cours/SLAM%205/XML/images/microsoft_xmlnotepad.gif)**

Exemple : **Altova XMLSpy Enterprise Edition** ([**http://www.altova.com/download/xmlspy/xml\_editor\_enterprise.html**](http://www.altova.com/download/xmlspy/xml_editor_enterprise.html)), et **Altova XMLSpy Professional Edition** ([**http://www.altova.com/download/xmlspy/xml\_editor\_professional.html**](http://www.altova.com/download/xmlspy/xml_editor_professional.html)), le plus cher et le plus complet.

|  |  |
| --- | --- |
| **5** | Les DTD (Document Type Definition) : définition de la structure des balises. |

**Principe.**

* En **XML**, les balises ne sont pas prédéfinies ni figées (sauf quelques-unes pour la structure globale). On peut créer nos propres balises, c'est la notion d'**eXtensibilité** du langage. Cela permet de leur associer une sémantique particulière, par exemple on devine que la balise **<client> ... </client>**contient des informations sur le client d'une organisation. C'est ce qui fait que les documents XML sont compréhensibles même pour une personne novice.
* La **DTD** (**D**ocument **T**ype **D**éfinition, définition de type de document) permet de définir la structure de notre document XML, et notamment :
  + les balises qu'on est autorisé à utiliser ;
  + l'ordre d'apparition et la possibilité de répétition de ces balises ;
  + les attributs qu'on est autorisé à utiliser.

**Premier exemple appliqué au fichier "utilisateurs.xml".**

* Pour une première approche intuitive des DTD, commençons par créer ensemble la DTD de notre fichier **utilisateurs.xml**.

**<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>  
<utilisateurs>  
  <utilisateur type="Prof Info">  
    <nom>PASTORE</nom>  
    <prénom>Sébastien</prénom>  
    <mél>spastore2@free.fr</mél>  
    <date-naissance>1973-02-25</date-naissance>  
  </utilisateur>  
  <utilisateur type="903">  
    <nom>DUBUISSON</nom>  
    <prénom>Christophe</prénom>  
    <mél>christophe.dubuisson@laposte.net</mél>  
    <groupe>1</groupe>  
  </utilisateur> ….**

Résultat final :

<!DOCTYPE utilisateurs [

<!ELEMENT utilisateurs (utilisateur+)>

<!ELEMENT utilisateur (nom, prénom, mél?, groupe?, date-naissance?)>

<!ELEMENT nom (#PCDATA)>

<!ELEMENT prénom (#PCDATA)>

<!ELEMENT mél (#PCDATA)>

<!ELEMENT date-naissance (#PCDATA)>

<!ELEMENT groupe (#PCDATA)>

<!ATTLIST utilisateur type CDATA #REQUIRED>

]>

**Intégration au fichier XML.**

La DTD peut être soit **interne** (incluse dans le fichier XML), soit **externe** (fichier séparé portant l'extension **.DTD**).

* Dans le cas de la DTD **interne**, voici comment se présente le fichier XML (remarquez qu'on utilise l'attribut **standalone="yes"**) :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <?xml version="1.0" standalone="yes"?>  <!DOCTYPE élément-racine [    ici, la DTD interne brute, par exemple :    <!ELEMENT élément-racine (...)>    ...    ]>      <élément-racine>      ...    </élément-racine> |

* Dans le cas de la DTD **externe**, voici comment se présente le fichier XML (remarquez qu'on utilise l'attribut **standalone="no"**, qui est d'ailleurs sa valeur par défaut) :

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>    <!DOCTYPE élément-racine SYSTEM "fichier.dtd">      <élément-racine>      ...    </élément-racine> |

Le fichier DTD contient, lui, la DTD brute. Par exemple, la [**DTD externe de mes cours**](http://cours.sebastien-pastore.com/cours/SLAM%205/XML/fichiers%20dtd/cours.dtd.txt).

* Voici un exemple de DTD brute, qu'elle soit interne ou externe. Remarquez d'une DTD n'est pas un document XML, elle ne contient donc pas l'entête**<?xml version="1.0" ?>** :

|  |  |
| --- | --- |
|  | <!ELEMENT information (auteur, texte) ><br>  <!ELEMENT auteur (#PCDATA) >  <!ELEMENT texte (#PCDATA) >  <!ATTLIST auteur type (salarie|intervenant) #REQUIRED > |

**Définition des balises.**

* + - Pour dire qu'une balise contient d'autres **balises filles**, dans un ordre précis :

|  |  |
| --- | --- |
|  | <!ELEMENT utilisateur (nom, prénom, mél, groupe, date-naissance)> |

*La balise****<utilisateur>****doit contenir les balises filles****<nom>****,****<prénom>****,****<mél>****,****<groupe>****,****<date-naissance>****, obligatoires et dans cet ordre.*

* + - Pour dire qu'une balise peut apparaître **une ou plusieurs fois**, on utilise l'opérateur **+** :

|  |  |  |
| --- | --- | --- |
|  | <!ELEMENT utilisateurs (utilisateur+)> | <!ELEMENT discotheque (disque+)> |

La balise **<utilisateurs>** contient une ou plusieurs fois la balise fille **<utilisateur>**.

* + - Pour dire qu'une balise peut apparaître **zéro, une ou plusieurs fois**, on utilise l'opérateur **\*** :

|  |  |  |
| --- | --- | --- |
|  | <!ELEMENT utilisateurs (utilisateur\*)> | <!ELEMENT discotheque (disque\*)> |

*La balise****<utilisateurs>****contient zéro, une ou plusieurs fois la balise fille****<utilisateur>****.*

* + - Pour dire qu'une balise peut apparaître **zéro ou une fois**, on utilise l'opérateur **?** :

|  |  |
| --- | --- |
|  | <!ELEMENT utilisateur (nom, prénom, mél?, groupe?, date-naissance?)>  *Les balises filles <mél>, <groupe> et <date-naissance> sont optionnelles.* |

* + - Pour dire qu'une balise doit contenir du **texte alphanumérique**, et pas de balises filles, on utilise le mot-clé **(#PCDATA)** (parsed-character data) :

|  |  |
| --- | --- |
|  | <!ELEMENT nom (#PCDATA)>  <!ELEMENT prénom (#PCDATA)>  <!ELEMENT mél (#PCDATA)> |

*Les balises <nom>, <prénom> et <mél> contiennent du texte alphanumérique.*

* + - Pour dire qu'une balise peut contenir **n'importe quoi** (**texte alphanumérique** ou **autres balises**), on utilise le mot-clé **ANY** :

|  |  |
| --- | --- |
|  | <!ELEMENT td ANY> |

*La balise <td> contient du texte quelconque ou n'importe quelle autre balise.*

* + - Pour dire qu'une balise est **vide** (par exemple les balises XHTML **<br>**, **<img>**, **<meta>**, etc.), on utilise le mot-clé **EMPTY** :

|  |  |
| --- | --- |
|  | <!ELEMENT img EMPTY> |

* + - On dispose également d'un opérateur *ou*, qui s'écrit **|** :

|  |  |
| --- | --- |
|  | <!ELEMENT tr (td|th)\*> |

*La balise <tr> contient une combinaison quelconque de balises filles <td> et <th>.*

Attention aux parenthèses :

**(A, B)+** : L'opérateur après la parenthèse s'applique au couple de balise **<A><B>**, ce qui autorise ce genre de choses : **<A><B><A><B><A><B>**.  
**(A, B+)** : L'opérateur dans la parenthèse ne s'applique qu'à la balise **<B>**, ce qui autorise ce genre de choses : **<A><B><B><B>**.

**Définition des attributs.**

* Qu'est-ce que c'est qu'un attribut ?  
    
  Par exemple, l'élément **utilisateur** contient l'attribut **type** :  
  **<utilisateur type="ProfInfo">**  
    
  Par exemple, la balise **img** contient les attributs obligatoires **src** et **alt** :  
  **<img src="images/fond.jpeg" alt="Image de fond" />**
* Dans une DTD, c'est la commande **<!ATTLIST ...>** qui nous permet de définir les attributs. La syntaxe générale est :  
  **<!ATTLIST *nom-élément nom-attribut type défaut*>**  
    
  On définit, en un seul **ATTLIST**, tous les attributs d'une même balise. Effectivement, **ATTLIST** signifie "**Liste d'attributs**" :  
  **<!ATTLIST *nom-élément  
      nom-attribut1 type1 défaut1  
      nom-attribut2 type2 défaut2*  
      nom-attribut3 type3 défaut3>**
* La valeur du *pour type* peut être (les plus utiles en premier) :
  + **CDATA**: valeur alphanumérique quelconque (par exemple pour **width="100"**)
  + **(*Valeur1*| *Valeur2*|*...*)** : valeur au choix (par exemple pour **align="right"**)
  + **ID**: valeur unique pour chaque élément, comme une clé primaire.
  + **IDREF**: référence un attribut de type **ID** (comme une clé étrangère).
  + etc.
* La valeur pour *défaut* peut être (non exhaustif, que les plus utilisés) :
  + **"*valeur par défaut*"** : dans le cas d'un attribut facultatif, si l'attribut est omis alors cette valeur par défaut est utilisée.
  + **#REQUIRED** : attribut obligatoire. Par exemple :  
    **<!ATTLIST input type #REQUIRED>**
  + **#IMPLIED** : attribut facultatif, s'il est omis alors il n'a pas de valeur par défaut. Par exemple :  
    **<!ATTLIST font color #IMPLIED>**
* Par exemple, pour la balise **<img>** :   
  **<!ATTLIST img  
      src CDATA #REQUIRED  
      alt CDATA #REQUIRED  
      align (top|middle|left|right) "top"  
      width CDATA #IMPLIED  
      height CDATA #IMPLIED  
      border CDATA "1">**

Récupérez ce fichier : [**discotheque.xml**](http://cours.sebastien-pastore.com/cours/SLAM%205/XML/fichiers%20xml/discotheque.xml).

Créez-lui sa DTD interne, puis copiez cette DTD ici ; idem avec la DTD externe.