

XML + XSL

1. Le langage XML

1.1. Avant de débiter ...

Prérequis

Si le langage Html est accessible au plus grand nombre, avec le langage XML vous jouez déjà un peu dans "la cour des grands". Le XML est de loin plus abstrait et donc plus complexe que le Html. Bien que ce tutorial se limitera à une découverte basique du XML, il est quasi indispensable pour en tirer quelques profits d'avoir :

- une connaissance et une pratique aigüe du langage Html.
- une connaissance et une pratique de la conception de pages Web.
- de bonnes notions de feuilles de style (CSS).
- des notions de Javascript ou de VBscript.

Le XML, en lui-même, ne fait rien !

Alors que le Html a été conçu pour afficher de l'information, le XML a été créé pour structurer de l'information. Il ne fait rien d'autre !

Voici un exemple de XML.

```
<?xml version="1.0"?>
<demoXML>
<message>Voici du XML</message>
</demoXML>
```

Ce qui affiché dans le Internet Explorer donne le résultat suivant.

```
<?xml version="1.0" ?>
- <demoXML>
  <message>Voici du XML</message>
</demoXML>
```

Pas que quoi fouetter un chat sur le plan esthétique... Le XML n'est que de l'information encodée entre des balises. Il faudra d'autres éléments, comme par exemple un fichier XSL, pour que le navigateur puisse "comprendre" vos balises et afficher ce fichier sous une forme plus conviviale. D'où notre titre : XML plus XSL ou XML + XSL.



Les limites et les objectifs

Le XML est un langage de professionnels de la conception de sites et ne sera que très rarement utilisé par les amateurs, même éclairés, de la publication sur le Web auxquels s'adresse ce site. Que ces amateurs soient cependant rassurés, pour eux le Html a encore de beaux jours devant lui... Mais pour les "pros" du Web, dès qu'il s'agira de stoker, traiter, envoyer des données, le XML sera la voie informatique royale de l'avenir.

Le XML est un métalangage soit un langage pour écrire d'autres langages. Ici aussi, il n'y a que peu de chances que vous conceviez un jour votre propre langage ! Mais le XML est une véritable révolution dans le panorama des langages de publication sur le Web. Il apparaît comme incontournable car il est déjà à la base de toute une série de nouveaux langages qui sont ou qui seront utilisés dans la conception des pages Internet comme le XHTML, le successeur désigné du Html, le WML pour le Wap des téléphones mobiles, le MathML pour les mathématiques, le SOAP et à n'en pas douter bien d'autres encore. Ces nouveaux langages générés par le XML en reprennent l'esprit, les règles et la syntaxe que vous pouvez découvrir ici.

1.2. Le langage XML

Le SGML

Le SGML pour *Standard Generalized Markup Language* est un langage normalisé pour la génération de langages de balises. Cette norme internationale [ISO8879] pour décrire la structure et le contenu de différents types de documents électroniques remonte à la nuit des temps de l'informatique et d'Internet. Ce langage très professionnel a la particularité d'être très concis et très abstrait. En conséquence, il n'est que très difficilement utilisable par le commun des mortels. Sa descendance est pourtant assez nombreuse et vous ne pouvez pas connaître un de ses enfants qui est un langage de balises utilisé pour la publication sur le Web : le HTML *HyperText Markup Language*.

Le Html ayant mal vieilli au fil des versions, le W3C consortium qui tente de régir les règles de la publication sur le Web, a décidé de repartir d'une feuille blanche en revenant en quelques sortes aux sources. D'où le XML eXtensible Markup Language, qui, outre le fait d'être issu du SGML, présente de fortes similitudes avec celui-ci. Ainsi, le XML peut-être considéré comme un SGML simplifié ou abrégé, un SGML qui serait abordable par le commun des webmestres.

Le XML serait plutôt un SGML^{- -} qu'un HTML^{+ +}.

Le XML

Le XML pour *eXtensible Markup Language* est donc un langage de balises comme le Html mais il est extensible, évolutif. En XML, les balises ne sont pas prédéfinies. C'est vous qui devez ou pouvez définir vos propres balises.

Et c'est là le problème ! Si les braves navigateurs n'avaient plus de difficultés pour afficher les balises prédéfinies du Html comme les <H1>,
 ou autres <TABLE>, que doivent-ils faire avec vos balises <ok> ou <new> ? Le XML a comme vocation de décrire de l'information et pas d'afficher celle-ci. Ainsi le XML pourtant créé en 1999, est resté durant près de deux ans, un concept plutôt abstrait et théorique faute de moyens fiables pour en afficher le résultat. Avec le développement de nouvelles techniques comme le XSL, il est devenu possible de percevoir concrètement les énormes potentialités de ce nouveau langage.

L'avenir du XML

Si le Html a régné en maître sur le Web durant la dernière décennie du 20ème (1990 à 2000), le XML sera, sans aucun doute possible, le standard omniprésent pour tout ce qui concerne la manipulation et la transmission des données durant la première décennie du 21ème siècle. Mais au risque ne me répéter, ce n'est pas tant le XML lui-même que vous utiliserez mais surtout les nombreux langages qui en découleront.

Seul bémol à ces prédictions euphoriques, est la relative inertie des navigateurs grands publics à permettre, courant 2001, l'exploitation de ces nouvelles prescriptions. Comme exemple on peut citer le MathML qui est un langage issu du XML et qui a pour vocation d'afficher les formules mathématiques. Ce langage bien que parfaitement défini n'est encore exploitable que par un seul navigateur, assez confidentiel, Amaya pour pas le nommer.

1.3. Le XML et le HTML

Le HTML et le XML ne sont pas comparables

Lorsqu'on étudie les moyens de publication sur le Web, on est inévitablement tenté de faire une comparaison entre le HTML et le XML. Au contraire de ce qui a déjà été écrit par ailleurs, le XML n'est pas le successeur du Html. Le XML n'est pas le futur du Html. Le XML n'est pas le remplaçant du Html.

Le XML et le HTML sont deux langages distincts !
--

Une seule similitude : le SGML

Le seul point commun entre le HTML et le XML est qu'ils sont issus tous deux de la même "mère" soit le SGML Standardized Generalised Markup Language qui est le langage de référence en milieu professionnel pour tout ce qui concerne la gestion électronique des documents. Ils sont donc, tous deux, des langages de balises [Markup Language]. Ils ont également des caractéristiques communes héritées du SGML qui sont de transporter sur le Web des données en mode texte [plain text], compatibles avec n'importe quelle plateforme logicielle.

Le XML et le HTML
Ils sont tous deux issus du SGML avec lequel ils partagent des caractéristiques communes
<ul style="list-style-type: none">◦ Ils fonctionnent avec des balises.◦ Ils sont indépendants de la plateforme.◦ Ils sont en mode texte [plain text].

Les différences entre le HTML et le XML

Le HTML et le XML sont différents en de très nombreux points dont certains ont trait à l'essence même du langage.

- Le XML décrit, structure, échange des données tandis que le Html ne fait qu'afficher des données.

- Le XML est extensible et permet de créer ses propres balises en fonction des données traitées. En Html, les balises sont prédéfinies et donc figées.

Le XML

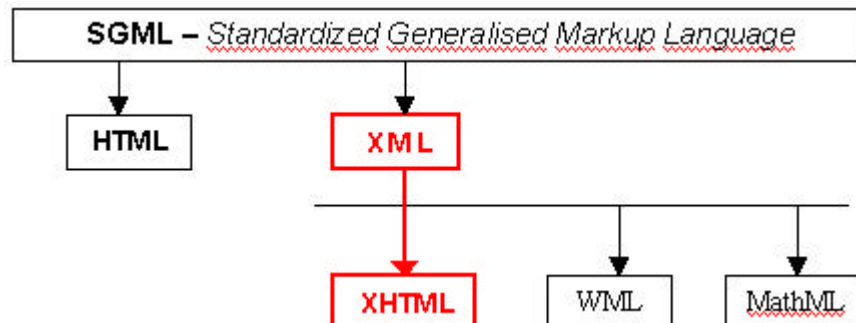
- Le XML décrit, structure, stocke, transporte et échange des données.
- Le XML est un générateur de langages [métalanguage].
- Outre les PCs, le XML se veut adapté aux outils comme les mobiles, les pockets, etc.
- Pour le XML, le W3C est reparti d'une feuille blanche et a mis en place un nouveau langage très structuré.
- Le XML est un langage strict dont l'écriture doit être rigoureuse.

Le HTML

- Le Html affiche des données par l'intermédiaire d'un navigateur.
- Le Html est un langage statique (normalisé) de publication sur le Web.
- Le Html est surtout conçu pour les ordinateurs de type PC.
- Le Html avec la version 4.0 est arrivé à bout de course et est devenu un langage hybride et en final peu structuré..
- Le Html, à cause des navigateurs récents, est devenu très permissif.

HTML, XML et XHTML ...:

Le XHTML est quant à lui le successeur du Html. Mais il est par ailleurs aussi un des "enfants" engendrés par le XML. En deux mots, pour faire un peu le ménage dans les dérives du Html au fil des différentes versions, le W3C a conçu le XHTML qui n'est en fait qu'une reformulation du HTML 4.0 selon la syntaxe et les règles du XML.



1.4. La syntaxe du XML

Le XML impose des règles de syntaxe très spécifiques par rapport au Html. En outre, on retrouvera ces mêmes règles de syntaxe dans tous les langages dérivés du XML comme le XHTML ou le WML par exemple.

- Le XML est un langage de balises [Markup Language].

Mais au contraire du Html où les balises sont définies, vous devez inventer vos balises. Rappelez-vous, le XML est eXtensible. Il faut donc écrire soi-même le nom des balises utilisées.

Il y a quelques règles pour la composition des noms (mais elle ne déroutent pas les habitués du Javascript) :

- Les noms peuvent contenir des lettres, des chiffres ou d'autres caractères.

- Les noms ne peuvent débuter par un nombre ou un signe de ponctuation.
- 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 comme "-", ";", ".", "<", ">", etc.
- Les caractères spéciaux pour nous francophones comme é, à, ê, ï, ù sont à priori permis mais pourraient être mal interprétés par certains programmes.

On profitera de cette liberté dans les noms pour les rendre le plus descriptif possible comme par exemple <gras_et_italique>.

- Les balises sont sensibles au majuscules et minuscules [case sensitive].

Ainsi, la balise <Message> est différente de la balise <message>. La balise d'ouverture et la balise de fermeture doivent donc être identiques. Ainsi par exemple ; <Message> ... </message> est incorrect et <message> ... </message> est correct.

Une tendance se dégage pour n'écrire les balises qu'en minuscules, limitant ainsi les erreurs possibles.

- Toute balise ouverte doit impérativement être fermée.

Finis les écritures bâclées du Html où l'on pouvait dans certains cas omettre la balise de fin comme pour le paragraphe <p> ou l'élément de liste .

Ainsi en Html, ce qui suit est affiché correctement :

```
<p>
<ul>
<li>Point 1
<li>Point 2
```

Le XML est beaucoup plus strict. On devrait avoir :

```
<p>
<ul>
<li>Point 1</li>
<li>Point 2</li>
<p>
```

Les éventuelles balises uniques ou appelées aussi balises vides, comme
, <meta> ou en Html, doivent également comporter un signe de fermeture soit balise/. Ainsi une balise <meta/> est correcte en XML.

- Les balises doivent être correctement imbriquées.

Le XML étant très préoccupé par la structure des données, des balises mal imbriquées sont des fautes graves de sens.

Ainsi l'écriture suivante est incorrecte car les balises ne sont pas bien imbriquées :

```
<parent><enfant>Loïc</parent></enfant>
```

L'écriture correcte avec une bonne imbrication des éléments est :

<parent><enfant>Marine</enfant></parent>

- Tout document XML doit comporter une racine.

En fait, la première paire de balises d'un document XML sera considéré comme la balise de racine [root].

Par exemple :

```
<racine>
... suite du document XML ...
</racine>
```

Si on ose faire un lien avec le Html, votre élément racine était <body> ... </body>.

Tous les autres éléments seront imbriqués entre ces balises de racine.

Par exemple :

```
<parents>
  <enfants>
    <petits_enfants> ... </petits_enfants>
  </enfants>
</parents>
```

- Les valeurs des attributs doivent toujours être mises entre des guillemets.

Le XML peut avoir (comme le Html) des attributs avec des valeurs. En XML, les valeurs des attributs doivent obligatoirement être entre des guillemets, au contraire du Html où leur absence n'a plus beaucoup d'importance.

Ainsi, l'écriture suivante est incorrecte car il manque les guillemets.

```
<date anniversaire=071185>
```

La bonne écriture est :

```
<date anniversaire="071185">
```

Le XML est un langage strict. Votre document doit impérativement respecter la syntaxe du XML. On dira alors que le document est "bien formé" [Well-formed]. Seuls les documents "bien formés" seront affichés correctement. A la moindre erreur de syntaxe, le document ne sera pas ou ne sera que partiellement affiché.

1.5. Un premier document XML

Voici un premier document XML.

Rien de bien compliqué mais ce document sera étoffé en cours d'étude.

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

La déclaration <?xml version="1.0"?> indique au navigateur que ce qui suit est un document XML selon sa version 1.0. Vous remarquerez que cette balise ne comporte pas de signe de fermeture car cette balise n'est pas encore du XML.

On en profite généralement pour notifier le "character set" qui indique à

l'interpréteur XML [Parser] le jeu de caractères à utiliser. Le jeu de caractères "ISO-8859-1" a, pour nous francophones, l'avantage d'accepter la plupart des lettres avec des accents. Mais il existe d'autres jeux de caractères comme UTF-8 ou UTF-16 plutôt destinés aux anglo-saxons car ils ne reprennent pas les accents.

<racine>

L'élément racine indispensable au XML. Vous pouvez utiliser, à votre convenance, n'importe quel nom à l'intérieur de cette balise de racine.

... suite du document XML ...

Votre document XML proprement dit, qui respectera bien entendu scrupuleusement la syntaxe du XML ("bien formé").

</racine>

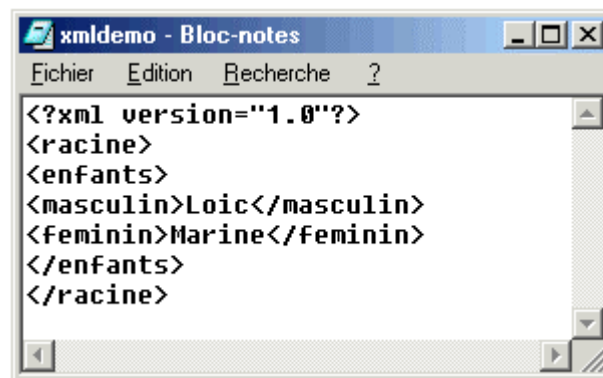
Le document XML se termine obligatoirement à la fermeture de la balise de racine.

Elaboration du fichier

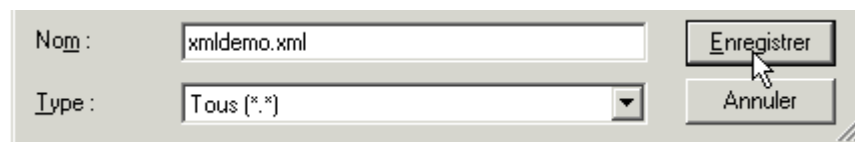
Voici un petit fichier XML.

```
<?xml version="1.0"?>
<racine>
<enfants>
<masculin>Loic</masculin>
<feminin>Marine</feminin>
</enfants>
</racine>
```

On le reproduit dans le programme Bloc-notes [notepad] pour les gens de Windows.

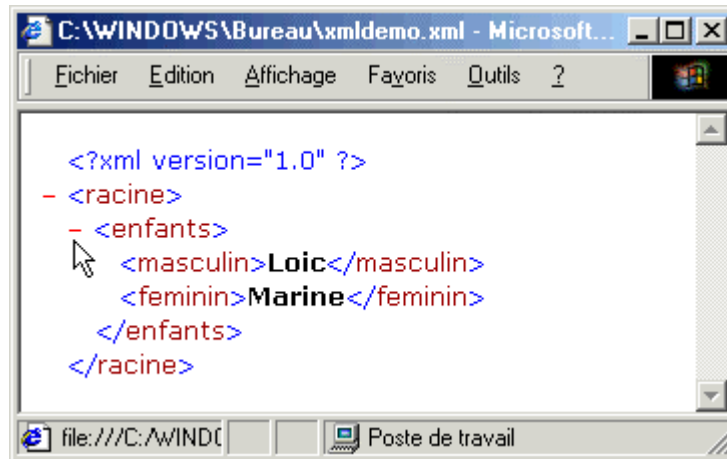


Et on l'enregistre (non pas en type de document Texte) en " Type : Tous (*.*)" " sous un nom avec une extension .xml.

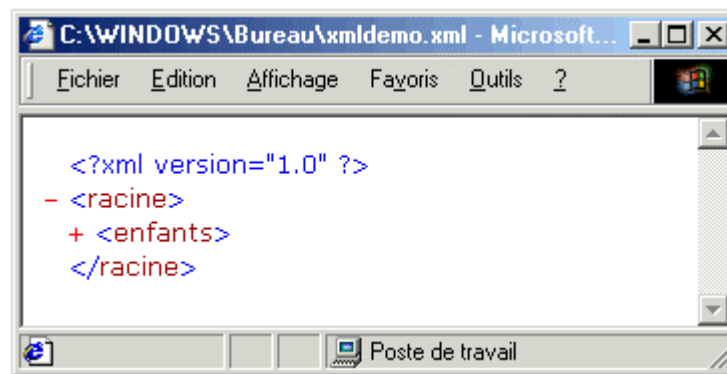


Résultat dans Microsoft Explorer 5 et +

Depuis le version 5 de Microsoft Internet Explorer, les fichiers XML s'affichent sans problèmes.

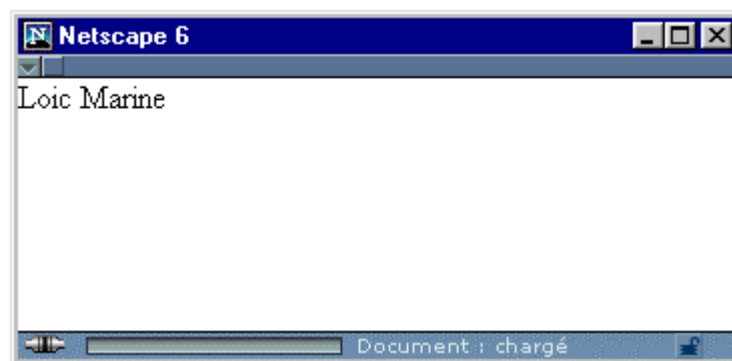


Vous remarquerez qu'il y a un petit signe - affiché devant des balises englobantes (voir le pointeur sur la capture d'écran). Il suffit de cliquer sur le signe pour masquer celles-ci. Et bien entendu de cliquer sur le signe + pour les faire réapparaître.



Résultat sous Netscape 6 et +

Le même fichier ne sera visible sur Netscape qu'à partir de la version 6. L'interprétation de ce fichier XML est pour le moins différente.



Au risque de me faire des ennemis, le XML et surtout le XSL est surtout l'affaire de Microsoft Explorer qui les prend mieux en compte. Espérons que ce ne soit que momentané.

1.6. Le DTD

Le Document Type Definition

Le DTD ou Document Type Declaration ou encore Document Type Definition est l'ensemble des règles et des propriétés que doit suivre le document XML. Ces règles définissent généralement le nom et le contenu de chaque balise et le contexte dans lequel elles doivent exister. Cette formalisation des éléments est particulièrement utile lorsqu'on utilise de façon récurrente des balises dans un document XML.

L'étude détaillée des DTDs dépassent de loin le cadre de cet ouvrage mais un bref aperçu est cependant utile surtout pour comprendre le fonctionnement des langages dérivés du XML qui ne manquent pas d'utiliser ces fameux DTDs.

En effet, par les DTDs externes, plusieurs concepteurs peuvent se mettre d'accord pour utiliser un DTD commun pour échanger leurs données. Avec le XHTML ou le WML, vous signalez dans l'en-tête du document que vous utilisez (et suivez) les normes du W3C concernant les langages précités.

Le DTD interne

On peut inclure son propre DTD au code source du fichier XML. On parlera alors d'un DTD interne.

Le DTD interne suit la syntaxe suivante :

```
<!DOCTYPE élément-racine [  
  déclaration des éléments  
>
```

Prenons un fichier comme exemple :

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

```
<!DOCTYPE parent [  

```

```
<!ELEMENT parent (garcon,filles)>
```

```
<!ELEMENT garcon (#PCDATA)>
```

```
<!ELEMENT fille (#PCDATA)>
```

```
>
```

```
<parent>
```

```
  <garcon>Loic</garcon>
```

```
  <filles>Marine</filles>
```

```
</parent>
```

Comme vous définissez un DTD interne, votre fichier est indépendant (standalone).

Début du DTD interne avec parent comme élément de racine.

Cet élément racine soit parent contiendra les sous-éléments garcon et fille.

#PCDATA indique au Parser XML que l'élément garcon contient des données exprimées en chiffres ou en lettres.

Idem pour l'élément fille.

Fin du DTD

Racine du document XML.

Fin du document XML.

Je ne peux résister à la tentation de livrer un extrait du DTD Strict pour la balise `` du XHTML.

```
<!ELEMENT img EMPTY>
```

```
  <!ATTLIST img
```

```
    %attrs;
```

La balise img est une balise vide [empty] dont les attributs sont

src	%URI;	#REQUIRED	src pour le lien (obligatoire) [required]
alt	%Text;	#REQUIRED	alt pour le texte alternatif (obligatoire)
longdesc	%URI;	#IMPLIED	longdesc
height	%Length;	#IMPLIED	height
width	%Length;	#IMPLIED	weight
usemap	%URI;	#IMPLIED	usemap
ismap	(ismap)	#IMPLIED	et ismap

>

Le DTD externe

Le DTD externe suivra la syntaxe suivante :

```
<!DOCTYPE élément-racine SYSTEM "nom_du_fichier.dtd">
```

Le même fichier que ci-dessus serait alors :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE parent SYSTEM "parent.dtd">
<parent>
  <garcon>Loic</garcon>
  <fille>Marine</fille>
</parent>
```

Le fichier de DTD externe (ici dans le même répertoire) "parent.dtd" contiendrait :

```
<!ELEMENT parent (garcon,fille)>
<!ELEMENT garçon (#PCDATA)>
<!ELEMENT fille (#PCDATA)>
```

Mais il est aussi possible de faire référence à un DTD externe situé sur un autre site comme pour par exemple le XHTML :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//FR"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Valide

Dans la littérature relative au XML, on distingue un document "bien formé" d'un document valide.

Un document valide est dit d'un document qui respecte les règles spécifiques de son DTD.

Un document "bien formé" est, pour rappel, un document qui respecte les règles générales de syntaxe du XML.

1.7. Afficher le XML avec CSS

Le CSS

Pour afficher les balises XML, on peut faire appel aux bonnes vieilles feuilles de style (CSS), maintenant classiques dans le paysage Html. A chaque balise "inventée" dans le fichier XML, on va définir un élément de style que le navigateur pourra alors afficher.

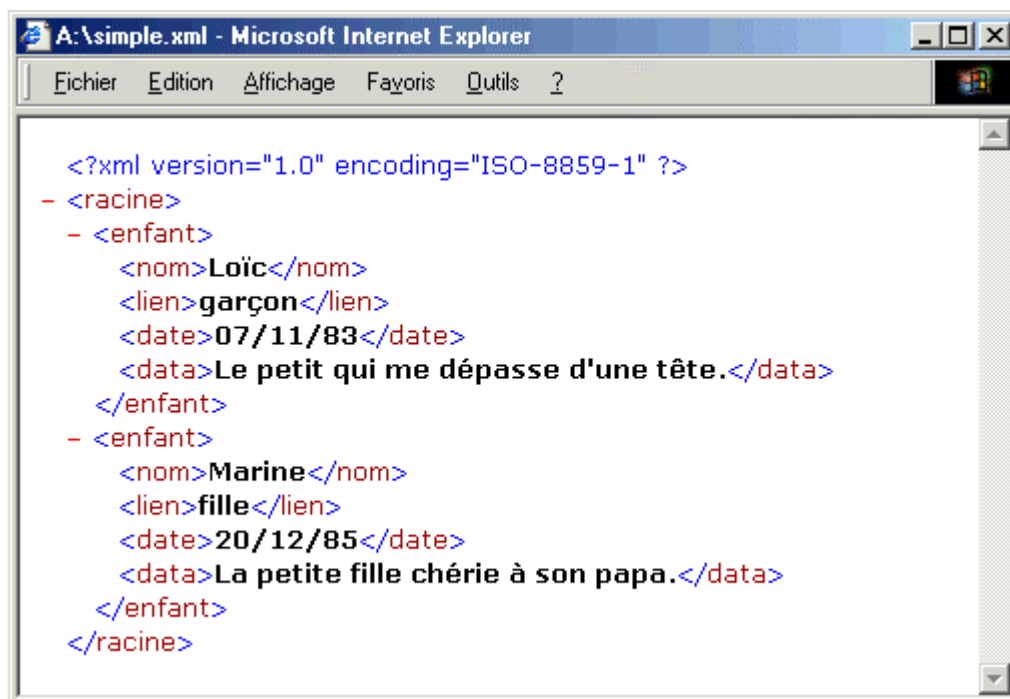
Un exemple de XML + CSS

A seule fin de démonstration, voici un exemple des possibilités d'une feuille de style CSS associée à un document XML.

Voici notre document XML de départ :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<racine>
  <enfant>
    <nom>Loïc</nom>
    <lien>garçon</lien>
    <date>07/11/83</date>
    <data>Le petit qui me dépasse d'une tête.</data>
  </enfant>
  <enfant>
    <nom>Marine</nom>
    <lien>fille</lien>
    <date>20/12/85</date>
    <data>La petite fille chérie à son papa.</data>
  </enfant>
</racine>
```

Affiché dans le navigateur, cela nous donne :



Tristounet !

On ajoute un fichier .css dont voici le contenu :

```
<style type="text/css">
```

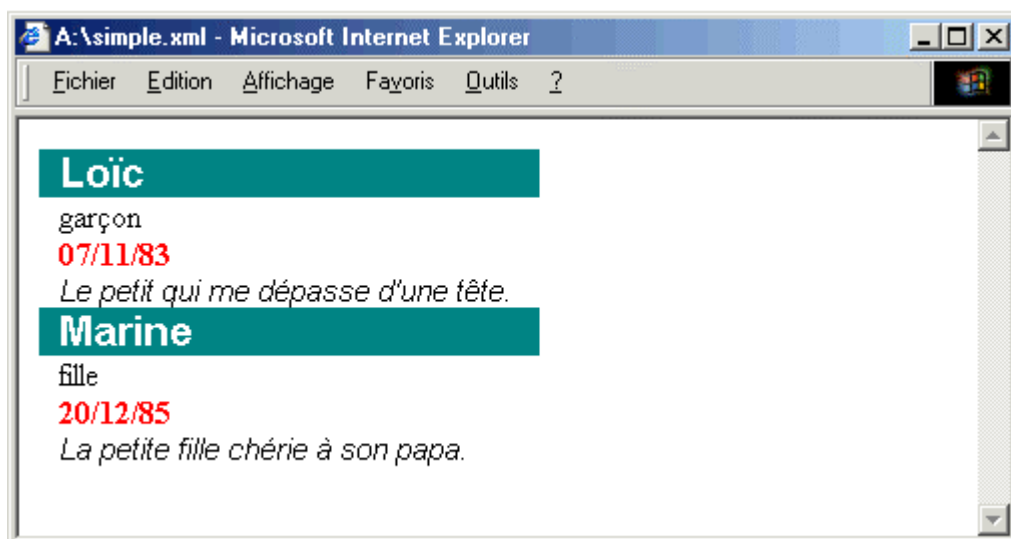
```

racine , enfant {
nom {
    display: block;
    width: 250px;
    font-size: 16pt ;
    font-family: arial ;
    font-weight: bold;
    background-color: teal;
    color: white;
    padding-left: 10px;
}
lien {
    display: block;
    font-size: 12pt;
    padding-left: 10px;
}
date {
    display: block;
    font-size: 12pt;
    color: red ;
    font-weight: bold;
    padding-left: 10px;
}
data {
    display: block;
    font-size: 11pt ;
    font-style: italic;
    font-family: arial ;
    padding-left: 10px;
}
}
</style>

```

Après avoir ajouté un lien vers le fichier css dans le fichier xml :

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



Plutôt sympa non ?...

Mais il y a encore un autre moyen, plus performant et aux possibilités plus étendues : afficher du XML avec le XSL soit le langage de feuilles de style eXtensible. Le pendant du XML au CSS.

1.8. Afficher le XML avec XSL

Le XSL - Les feuilles de style du XML

Comme le XML n'utilise pas des balises prédéfinies (car on peut inventer ses propres balises), le navigateur ne "comprend" pas les balises du XML et ne sais pas trop comment afficher un document XML.

Pour néanmoins afficher des documents XML, il est nécessaire d'avoir un mécanisme pour décrire comment le document pourrait être affiché. Un de ces mécanisme est les feuilles de style classiques du Html (CSS), mais le XSL pour eXtensible Stylesheet Language est de loin un langage de feuille de style plus adapté au XML et donc plus performant..



De façon résumée, le XSL est un langage qui transforme le XML en Html. Mais il fait bien plus ! Ainsi nous avons cru utile de lui consacrer un plus ample développement plus loin dans ce tutorial.

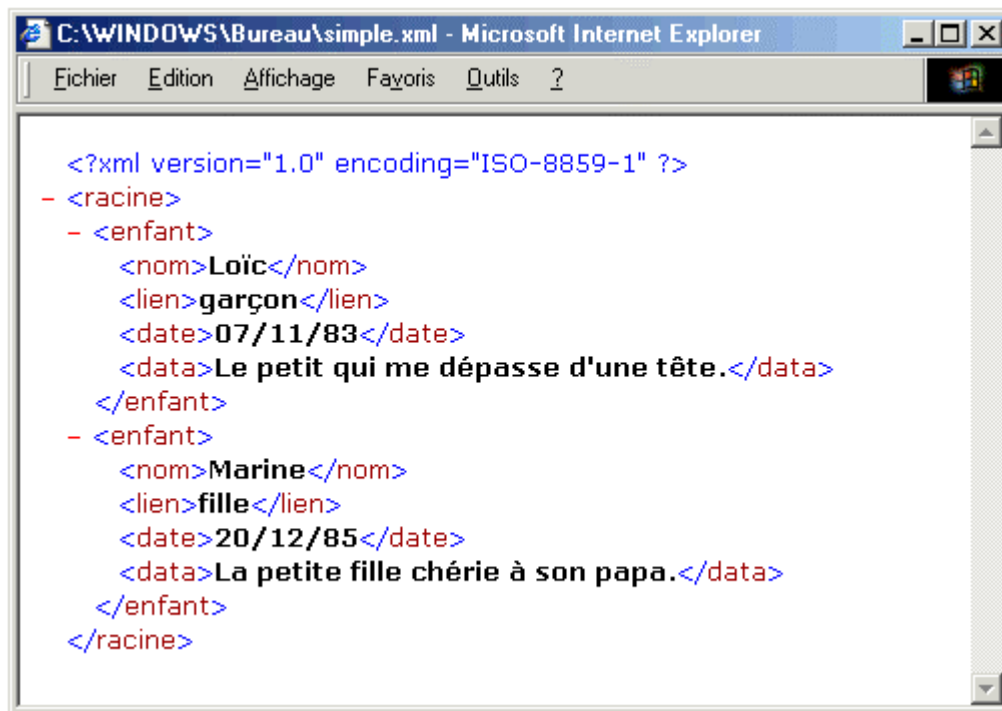
Un exemple de XML + XSL

A seule fin de démonstration, voici un exemple des possibilités du XSL associé à un document XML. Les explications seront données au chapitre consacré au XSL.

Voici notre document XML de départ :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<racine>
  <enfant>
    <nom>Loïc</nom>
    <lien>garçon</lien>
    <date>07/11/83</date>
    <data>Le petit qui me dépasse d'une tête.</data>
  </enfant>
  <enfant>
    <nom>Marine</nom>
    <lien>fille</lien>
    <date>20/12/85</date>
    <data>La petite fille chérie à son papa.</data>
  </enfant>
</racine>
```

Affiché dans le navigateur, cela nous donne :



Pas très folichon !

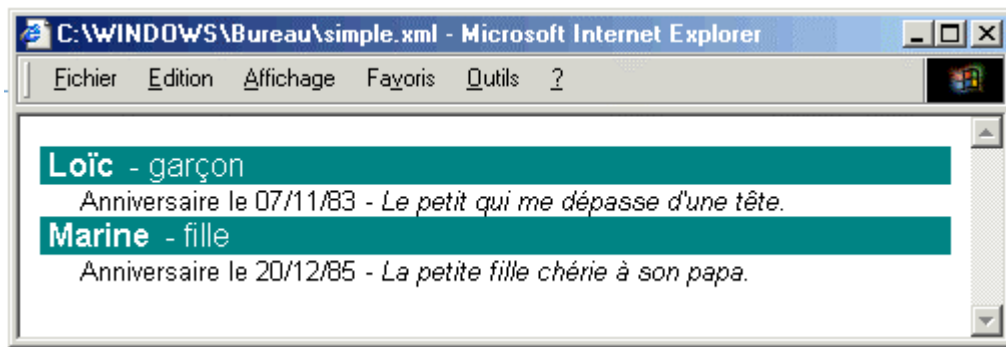
On ajoute un fichier .xsl dont voici le contenu :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<html xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<body style="font-family:Arial; font-size:12pt;">
  <xsl:for-each select="racine/enfant">
    <div style="background-color:teal; color:white;">
      <span style="font-weight:bold; color:white; padding:4px">
        <xsl:value-of select="nom"/></SPAN>
        - <xsl:value-of select="lien"/>
      </div>
      <div style="margin-left:20px; font-size:10pt">
        <span> Anniversaire le <xsl:value-of select="date"/>
        </span>
        <span style="font-style:italic"> - <xsl:value-of select="data"/>
        </span>
      </div>
    </xsl:for-each>
  </body>
</html>
```

Après avoir ajouté un lien vers le fichier xsl dans le fichier xml :

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

On obtient finalement :



Un peu mieux assurément !

1.9. Afficher du XML dans Html

Du XML dans un fichier Html

On peut toujours incorporer du XML dans un fichier Html avec la balise `<xml> ... </xml>`. Mais en toute logique, quand les navigateurs rencontrent des balises incorrectes ou inconnues, rien n'est affiché. Ce sera le cas avec vos balises XML incorporées dans un fichier Html. Heureusement, on peut passer par une astuce qui répond au doux nom romantique de "îlots de données" [Data Islands].

Les Data Islands [les îles de données]

Derrière ce nom pour le moins bizarre, se cache une possibilité assez intéressante. Dans un fichier Html, vous pouvez créer un "îlot" de données se trouvant dans un fichier XML distinct et en extraite des données que vous pouvez alors afficher dans le document Html.

Ici, dans le fichier Html, on va désigner le fichier xml extérieur avec un identifiant id :

```
<xml id="fichierxml" src="simple.xml"></xml>
```

Dans un tableau Html, que l'on relie par un attribut à la source des données au moyen de l'identifiant désigné plus haut [`datasrc="#id"`], on peut finalement aller reprendre des données du fichier XML avec l'attribut de champ de donnée qui a comme valeur le nom de la balise XML [`datafld="balise_xml"`].

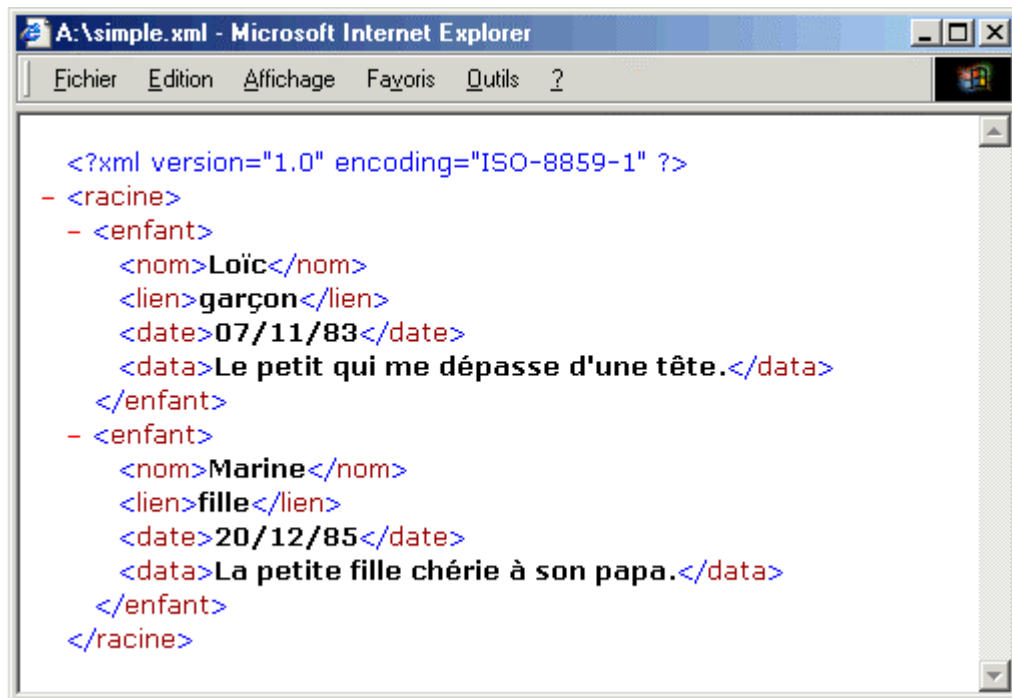
Vite, vite, un exemple !

Voilà toujours notre fichier XML (extérieur) :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<racine>
  <enfant>
    <nom>Loïc</nom>
    <lien>garçon</lien>
    <date>07/11/83</date>
    <data>Le petit qui me dépasse d'une tête.</data>
  </enfant>
  <enfant>
    <nom>Marine</nom>
    <lien>fille</lien>
    <date>20/12/85</date>
    <data>La petite fille chérie à son papa.</data>
  </enfant>
</racine>
```

```
</enfant>
</racine>
```

Soit :



Je vais créer un fichier Html classique dans lequel je voudrais reprendre des données du fichier XML et plus précisément le contenu des balises <nom>, <lien> et <date>.

```
<html>
<body>
Voici du Html...
<xml id="fichierxml" src="simple.xml"></xml>
<table border="1" datasrc="#fichierxml">
<tr>
<td><span datafld="nom"></span></td>
<td><span datafld="lien"></span></td>
<td>Anniversaire le <span datafld="date"></td>
</tr>
</table>
Et voici encore du Html !
</body>
</html>
```

Ce qui une fois affiché (avec quelques attributs supplémentaires pour le look de la page), offre le résultat suivant :



Grandiose !

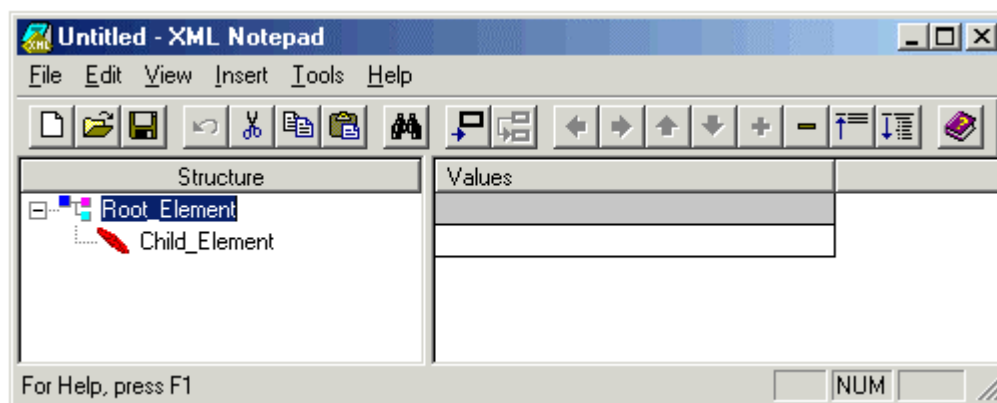
1.10. Des éditeurs XML expliqués

Les éditeurs...

N'attendez pas de miracle des éditeurs XML ! Comme en XML vous fabriquez sur mesure vos balises, les éditeurs ne peuvent avoir qu'un rôle d'aide à l'encodage et à la structure de votre document.

Les éditeurs XML sont cependant d'une grande utilité si vous avez de nombreuses balises récurrentes dans votre document XML. En outre, s'il en est nécessaire d'actualiser souvent les données de votre fichier Xml, il sera beaucoup plus facile de vous retrouver dans l'interface d'un éditeur que dans le fouillis de balises du code source.

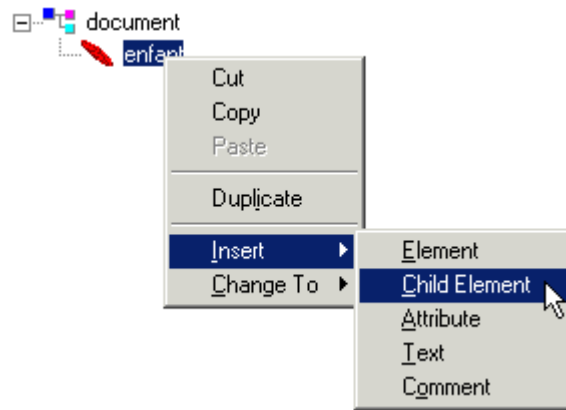
Microsoft XML Notepad



Nous allons reproduire le fichier XML qui nous a servi d'exemple jusqu'ici (simple.xml).

Cliquez sur "Root_Element" dans la fenêtre Structure pour l'élément racine et saisissez *document* au clavier. Cliquez ensuite sur "Child_Element" et encodez la balise *enfant*.

Ajoutons le sous-élément *nom*. Cliquez à cet effet sur l'élément *enfant*, avec le bouton droit de la souris et sélectionnez dans le menu contextuel Insert → Child Element.



Dans la fenêtre Structure, le curseur d'insertion clignote dans le sous-élément ajouté. On y ajoute la balise *nom*.

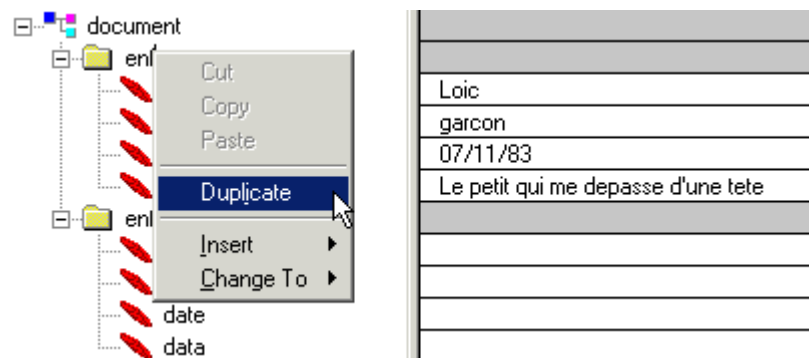
On accède ensuite au cadre de droite soit dans la fenêtre Valeur et on saisit *Loic*. A ma connaissance, XML Notepad ne permet pas de reprendre le jeu de caractères "ISO-8859-1".



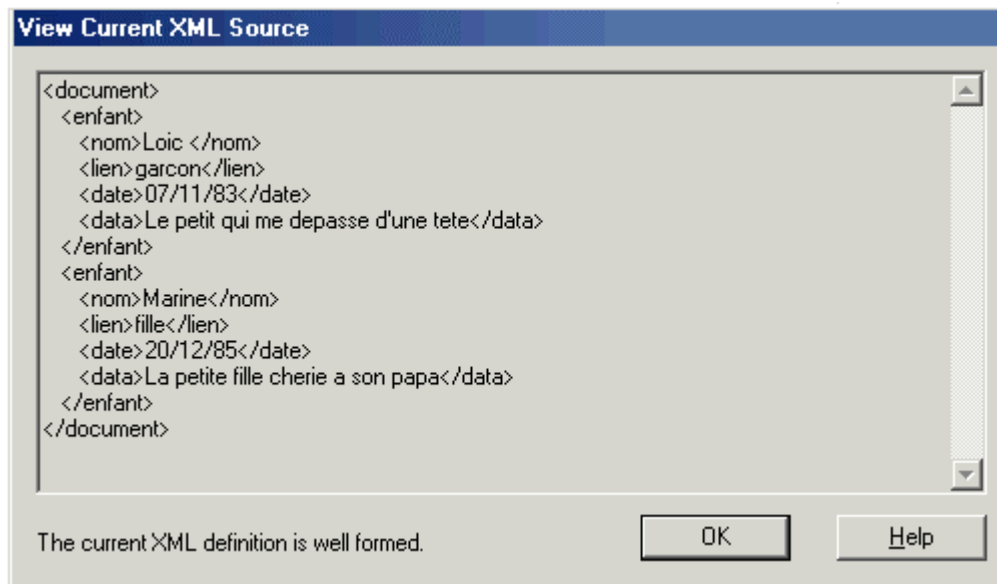
Ajoutons les autres sous-éléments *lien*, *date* et *data*. On clique sur *nom* → bouton droit → Insert → Element et on encode les données.



On va maintenant répéter la série de balises de sous-éléments. On clique sur *enfant* → bouton droit → Duplicate et la série est reproduite.

On peut encoder les valeurs.



Pour voir le document terminé, menu View → Source. En outre, dans la fenêtre ouverte, on vous signale si le document est "bien formé [well formed]".



Le bouton  est particulièrement utile pour trouver une donnée dans un fichier important tandis que la barre de direction  servira éventuellement à modifier aisément la structure du document XML.

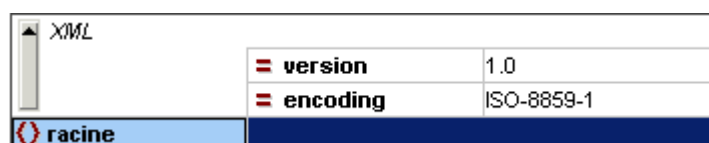
XML Spy

Avec XML Spy, on dispose d'un programme déjà plus professionnel. Il est à notre avis assez remarquable pour la structure qu'il met en place.

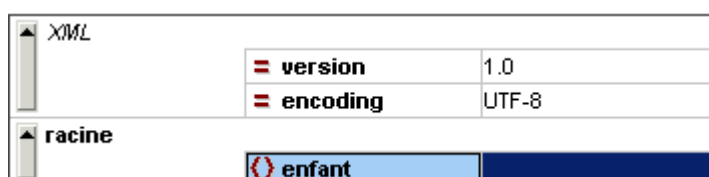


Nous allons reproduire le fichier XML qui nous a servi d'exemple jusqu'ici (simple.xml).

Commençons par l'élément racine. On prend le menu XML → Insert → Element et on encode la balise racine.



Encodons l'élément enfant. Après avoir cliqué dans la zone racine → menu XML → Add child → Element et on encode la balise enfant.



On passe aux sous-éléments. Après avoir cliqué dans la zone enfant, on demande 4 éléments soit menu XML → Add child → Element que l'on répète 4 fois et on remplit les balises et les données.

XML		
	version	1.0
	encoding	ISO-8859-1
racine		
	enfant	
	nom	Loic
	lien	garçon
	date	07/11/83
	data	Le petit qui me dépasse d'une tête.

Et l'on refait la même chose pour les autres données enfant.

XML		
	version	1.0
	encoding	ISO-8859-1
racine		
	enfant	
	nom	Loic
	lien	garçon
	date	07/11/83
	data	Le petit qui me dépasse d'une tête.
	enfant	
	nom	Marine
	lien	filie
	date	20/12/85
	data	La petite fille chérie à son papa.

Ceci n'est vraiment qu'un très bref aperçu de XML Spy dont les possibilités sont nettement plus nombreuses et étendues.

Xmetal

D'autres outils professionnels comme Xmetal de SoftQuad apparaissent sur le marché mais leur utilisation dépassent le cadre de ce tutorial.

2. Le langage XSL

2.1. Avant de débiter ...

Prérequis

Si le langage Html est accessible au plus grand nombre, avec le langage XML et XSL vous passez à une vitesse supérieure. Le XML et son complément le XSL est de loin plus abstrait et donc plus complexe que le Html. Bien que ce tutorial se limitera à une découverte basique du XSL, il est quasi indispensable pour en tirer quelques profits d'avoir :

- une connaissance basique du XML abordé dans le chapitre précédent.
- une connaissance et une pratique aigüe du langage Html.

- une connaissance et une pratique de la conception de pages Web.
- de bonnes notions de feuilles de style (CSS).
- des notions de Javascript ou de VBscript.

Le XML ne fait rien. Il faudra passer par le XSL !

Alors que le Html a été conçu pour afficher de l'information, le XML a été créé pour structurer de l'information. Il ne fait rien d'autre !

Voici un exemple de XML.

```
<?xml version="1.0"?>
<demoXML>
<message>Voici du XML</message>
</demoXML>
```

Ce qui affiché dans le Internet Explorer donne le résultat suivant.

```
<?xml version="1.0" ?>
- <demoXML>
  <message>Voici du XML</message>
</demoXML>
```

Le XML n'étant que de l'information encodée entre des balises, il faudra donner au navigateur d'autres éléments pour qu'il puisse "comprendre" vos balises et afficher ce fichier sous une forme plus conviviale. C'est là le rôle du XSL que nous étudierons ci-après.

Le XSL est donc le complément indispensable pour l'affichage du XML. D'où notre titre : XML plus XSL ou XML + XSL.



Reprenons notre fichier XML et associons lui un fichier (externe) XSL :

```
<?xml version="1.0"?>
<?xml-stylesheet href="fichierxsl.xml"?>
<demoXML>
<message>Voici du XML</message>
</demoXML>
```

Voici le fichier XSL :

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<html>
```

```

<body>
<xsl:value-of select="demoXML/message"/>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

Le résultat dans le navigateur est alors :

Voici du XML.

Beaucoup de travail et donc d'encodage pour un maigre résultat. Oh que non car la richesse des feuilles de style permettra de donner à l'affichage toute sa splendeur.

Le XSL ne fait pas que cela !

Le XSL ne permet pas uniquement l'affichage de XML. Il permet aussi :

- de sélectionner une partie des éléments XML.
- de trier des éléments XML.
- de filtrer des éléments XML en fonction de certains critères.
- de choisir des éléments.
- de retenir des éléments par des tests conditionnels.

2.2. Le langage XSL

Le XSL

Le XSL pour *eXtensible Stylesheet Language* ou "langage extensible de feuilles de style" est une recommandation du W3C datant de novembre 1999. C'est donc un standard dans le domaine de la publication sur le Web. Le XSL est en quelque sorte le langage de feuille de style du XML. Un fichier de feuilles de style reprend des données XML et produit la présentation ou l'affichage de ce contenu XML selon les souhaits du créateur de la page.

Le XSL comporte en fait 3 langages :

- Le **XSLT** qui est un langage qui **T**ransforme un document XML en un format, généralement en Html, reconnu par un navigateur.
- Le **Xpath** qui permet de définir et d'adresser des parties de document XML.
- Le **XML Formatter** pour "formater" du XML (transformé) de façon qu'il puisse être rendu sur des PCpockets ou des unités de reconnaissance vocale.

Pour la suite de ce tutorial, nous nous limiterons au XMLT et Xpath. Et comme dans la littérature relative à ce sujet, nous reprendrons le tout sous le terme général de XSL.

Le XSL est dérivé du XML

Le langage XML est un langage de balises dérivé du langage XML. Le XSL reprend donc toutes les règles de syntaxe du XML (détaillée dans la partie relative au XML).

Reprenons en bref :

- les balises sensibles à la casse, s'écrivent en minuscules.
- toutes les balises ouvertes doivent être impérativement fermées.
- les balises vides auront aussi un signe de fermeture soit <balise/>.
- les balises doivent être correctement imbriquées.
- les valeurs des attributs doivent toujours être mises entre des guillemets.
- le document XSL devra être "bien formé" [Well-formed].

Quel navigateur ?

A l'heure actuelle (mi-2001), seul Microsoft Internet Explorer depuis sa version 5 reconnaît le XML. Les exemples de ce tutorial ne fonctionneront donc que si vous utilisez Internet Explorer 5 ou plus.

Il faut préciser que Le XML des versions 5 et 5.5 n'est pas compatible à 100% avec la dernière recommandation du W3C. Pour la petite histoire, Explorer 5 est apparu alors que le XSL n'était encore qu'au stage de projet ou de document de travail du W3C [working draft].

Le correcteur syntaxique XML [essai de traduction de "XML Parser"] s'est affiné au fil des versions de Microsoft Internet Explorer :

- MSXML 2.0 est la référence du XML Parser de Internet Explorer 5.0 (mais pas compatible à 100%).
- MSXML 2.5 est la référence du XML Parser de IE 5.5 (en forte amélioration sans être parfait cependant).
- MSXML 3.0 est la dernière version du XML Parser. MSXML 3.0 peut déjà être téléchargé à partir du site de Microsoft et devrait être implémenté dans Internet Explorer 6.0. A en croire les gens de Microsoft MSXML 3.0 et donc Internet Explorer 6.0 serait à 100% compatible avec les spécifications XML et XSL du W3C.

2.3. Un premier document XSL

Principe de fonctionnement

Avant de débiter, il est utile de préciser :

1. que le XSL est dérivé du XML. Le document XSL reprend donc la structure et la syntaxe de n'importe quel document XML.
2. que le document XSL comporte un document Html ou Xhtml qui sera quant à lui reconnu par le navigateur et qui servira de support à tout ou partie des données du document XML associé.
3. que le XSL fonctionne avec une ou plusieurs "templates", sorte de gabarit pour définir comment afficher des éléments du fichier XML. Les éléments concernés du fichier XML sont déterminés par l'attribut "match".

Voici un premier document XSL.

Rien de bien compliqué mais ce document sera étoffé en cours d'étude.

<?xml version="1.0"?>

Le XSL est dérivé du XML. Il est normal que le document XSL commence par la déclaration de document XML, soit <?xml version="1.0"?>.

<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

La seconde ligne déclare que le document est du XSL extensible stylesheet.

L'attribut xmlns fait référence au "namespace" utilisé. Le namespace officiel du W3C est xmlns:xsl="http://www.w3.org/1999/XSL/Transform". Pour la petite histoire 1999 fait référence à l'année d'apparition du concept XSL. Le xmlns (incorrect) de Microsoft IE soit xmlns:xsl="http://www.w3.org/TR/WD-xsl"> est dû au fait que le XSL a été implanté dans Internet Explorer 5.0 alors qu'il n'était encore qu'en cours d'élaboration [working draft] par le W3C.

<xsl:template match="/">

Voilà une balise template et son attribut match.

Cette balise template va déterminer un gabarit dans lequel on va transformer des éléments du fichier XML sous une forme que le navigateur pourra afficher.

Les éléments du fichier XML sont déterminés par l'attribut match="/". Le slash / entre guillemets signale que sont concernées toutes les balises XML du document associé à partir de la racine [root].

<html>

<body>

Début de la partie Html qui servira de support pour l'affichage du document dans le navigateur. Attention, balises en minuscules !

Diverses balises Html et XSL...

Par exemple :

<xsl:value-of select="chemin d'accès/élément"/>

La balise <xsl:value-of> sera fréquemment utilisée car elle permet de sélectionner un élément du fichier XML associé pour le traiter dans le fichier XSL. Dans l'attribut select, on détermine le chemin d'accès vers la balise XML souhaitée (puisque le XML est structuré) comme le chemin d'accès de répertoire en sous-répertoire vers un dossier. Attention, on utilise bien ici le "forward slash" soit / .

</body>

</html>

Fin de la partie en Html.

</xsl:template>

La fermeture de la balise de template.

</xsl:stylesheet>

Le document XSL se termine obligatoirement par la fermeture de la balise de déclaration de document XSL.

Attention ! Pour que ce fichier XSL soit d'une quelconque utilité, il faut encore faire référence **dans le fichier XML**, au fichier XSL.

On ajoutera donc dans le fichier XML :

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

Cette balise indique au navigateur qu'une feuille de style [stylesheet] est associé au fichier XML et qu'il doit aller chercher le fichier de style à l'adresse indiquée par l'attribut href.

2.4. Un premier exemple XSL

Elaboration du fichier

Après cet aperçu théorique, étudions un exemple détaillé, soit une compilation de fichiers MP3.

Voici un fichier XML que l'on reproduit dans le Bloc-notes ou Notepad pour les "Windows"



```
<?xml version="1.0"?>
<compilation>
  <mp3>
    <titre>Foule sentimentale</titre>
    <artiste>Alain Souchon</artiste>
  </mp3>
  <mp3>
    <titre>Solaar pleure</titre>
    <artiste>MC Solaar</artiste>
  </mp3>
  <mp3>
    <titre>Le baiser</titre>
    <artiste>Alain Souchon</artiste>
  </mp3>
  <mp3>
    <titre>Pourtant</titre>
    <artiste>Vanessa Paradis</artiste>
  </mp3>
  <mp3>
    <titre>Chambre avec vue</titre>
    <artiste>Henri Salvador</artiste>
  </mp3>
</compilation>
```

Que l'on enregistre (non pas en type de document Texte) en " Type : Tous (*.*)" sous le nom xmldemo avec une extension .xml.

Nom : xmldemo.xml

Type : Tous (*.*)

Enregistrer

Annuler

Passons maintenant au fichier XSL



Le but de l'exercice est de représenter la compilation de mp3 sous forme d'un tableau.

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
  <html>
  <body>
```

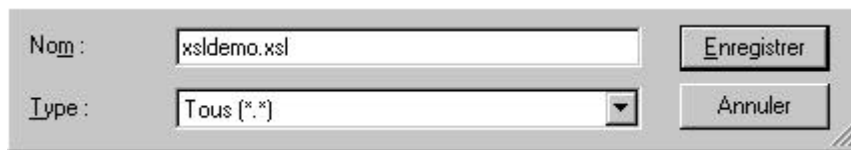
```

<table border="1" cellspacing="0" cellpadding="3">
<tr bgcolor="#FFFF00">
<td>Titre</td>
<td>Artiste</td>
</tr>
<tr>
<td><xsl:value-of select="compilation/mp3/titre"/></td>
<td><xsl:value-of select="compilation/mp3/artiste"/></td>
</tr>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

Après les balises de départ d'un fichier XSL, on aborde un tableau tout à fait classique en Html. On remplit la cellule du titre par la balise **xsl:value-of** avec l'attribut **select="compilation/mp3/titre"** qui indique comme chemin d'accès la balise racine compilation → la balise mp3 → la balise titre. Et bien entendu de même pour la balise artiste.

On enregistre en " Type : Tous (*.*) " sous le nom xsldemo avec une extension .xsl.



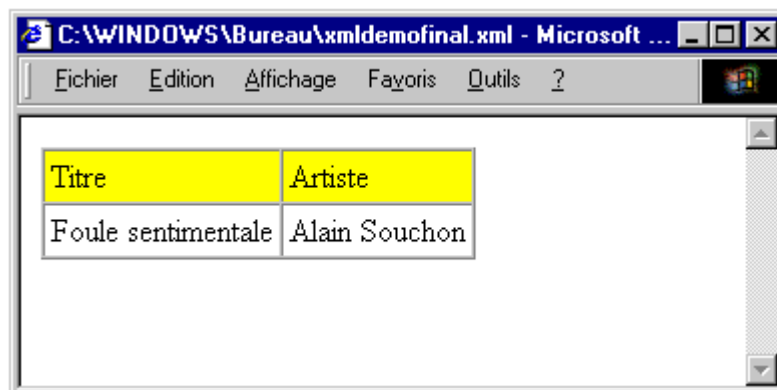
On revient au fichier XML et on y ajoute la balise pour y associer le fichier XSL.

```

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="xsldemo.xsl"?>
<compilation>
<mp3>
<titre>Foule sentimentale</titre>
<artiste>Alain Souchon</artiste>
</mp3>
etc...

```

Et miracle, notre stupide fichier XML plein de balises devient un beau tableau sympathique.



Afficher toutes les données

J'en vois là-bas qui sont un peu déçus de n'avoir qu'une référence de la compilation affichée dans le tableau. Bien allons-y...

Le fichier XML ne change pas. Soit :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="xsldemo?.xsl"?>
<compilation>
  <mp3>
    <titre>Foule sentimentale</titre>
    <artiste>Alain Souchon</artiste>
  </mp3>
  <mp3>
    <titre>Solaar pleure</titre>
    <artiste>MC Solaar</artiste>
  </mp3>
  <mp3>
    <titre>Le baiser</titre>
    <artiste>Alain Souchon</artiste>
  </mp3>
  <mp3>
    <titre>Pourtant</titre>
    <artiste>Vanessa Paradis</artiste>
  </mp3>
  <mp3>
    <titre>Chambre avec vue</titre>
    <artiste>Henri Salvador</artiste>
  </mp3>
</compilation>
```

On va reprendre le fichier XSL auquel on va ajouter la balise **xsl:for-each** [pour chaque] avec comme attribut **select="compilation/mp3"**. Pour chaque arborescence où l'on retrouve les balises compilation et mp3, il suffit de faire une ligne de tableau **<tr>** avec dans la premier cellule **<td>**, le contenu de la balise **<titre>** **xsl:value-of select="titre"** et dans la seconde cellule **<td>**, le contenu de la balise **<artiste>** **xsl:value-of select="artiste"**.

Ce qui donne :

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <html>
      <body>
        <table border="1" cellspacing="0" cellpadding="3">
          <tr bgcolor="#FFFF00">
            <td>Titre</td>
            <td>Artiste</td>
          </tr>
          <xsl:for-each select="compilation/mp3">
            <tr>
              <td><xsl:value-of select="titre"/></td>
```

```

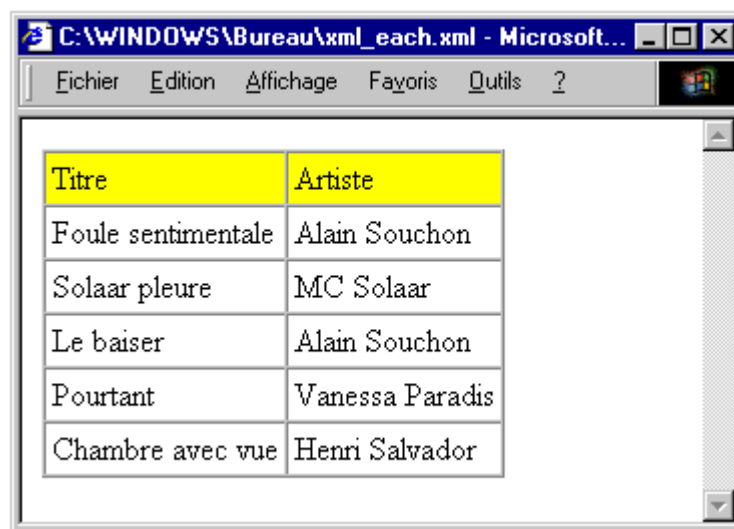
<td><xsl:value-of select="artiste"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

On enregistre le fichier avec l'extension .xsl.

On n'oublie surtout pas de modifier le lien dans le fichier XML de base.

Il ne reste plus qu'à admirer le fichier dans le navigateur.



Titre	Artiste
Foule sentimentale	Alain Souchon
Solaar pleure	MC Solaar
Le baiser	Alain Souchon
Pourtant	Vanessa Paradis
Chambre avec vue	Henri Salvador

La suite..?

Le langage XSL est plus qu'une série de balises pour afficher du XML. Il comporte aussi des possibilités plus qu'utiles quand on est confronté à des données. Nous verrons plus loin comment le XSL permet :

- de trier les données XML en ordre croissant ou décroissant.
- de filtrer des éléments XML en fonction de certains critères.
- de choisir des éléments.
- de retenir des éléments par des tests conditionnels.

2.5. Trier avec le langage XSL

Trier avec le XSL

Le langage XSL permet en quelques mots de trier des données du fichier XML associé en ordre croissant ou décroissant. Ainsi, il suffit d'ajouter l'attribut **order-by="+balise"** pour trier en ordre croissant et **order-by="-balise"** pour trier en ordre décroissant. Et c'est tout !

Il me plaît ici de souligner avec cette balise, la puissance du langage XSL. En outre si vous n'avez pas oublié les quelques mots d'anglais appris à l'école, le langage XSL est assez intuitif. Ainsi `order-by="+balise"` peut se lire : "ordonner ou trier par ordre croissant (+) les données comprises entre la balise désignée".

Elaboration du fichier

Reprenons notre fichier XML (inchangé).



```
<?xml version="1.0"?>
<compilation>
  <mp3>
    <titre>Foule sentimentale</titre>
    <artiste>Alain Souchon</artiste>
  </mp3>
  <mp3>
    <titre>Solaar pleure</titre>
    <artiste>MC Solaar</artiste>
  </mp3>
  <mp3>
    <titre>Le baiser</titre>
    <artiste>Alain Souchon</artiste>
  </mp3>
  <mp3>
    <titre>Pourtant</titre>
    <artiste>Vanessa Paradis</artiste>
  </mp3>
  <mp3>
    <titre>Chambre avec vue</titre>
    <artiste>Henri Salvador</artiste>
  </mp3>
</compilation>
```

Passons maintenant au fichier XSL



Nous allons trier notre compilation de mp3 en XML en ordre alphabétique croissant du nom des artistes. Et pour changer un peu, on permute les colonnes "titre" et "artiste" pour bien montrer que le XSL affiche les données du fichier XML selon le fichier Html (ou autre) qu'il contient.

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <html>
      <body>
        <table border="1" cellspacing="0" cellpadding="3">
          <tr bgcolor="#FFFF00">
            <td>Artiste</td>
            <td>Titre</td>
```

```

</tr>
<xsl:for-each select="compilation/mp3" order-by="+artiste">
<tr>
<td><xsl:value-of select="artiste"/></td>
<td><xsl:value-of select="titre"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

On enregistre le fichier sous le nom `xsl_order` avec une extension `.xsl`.

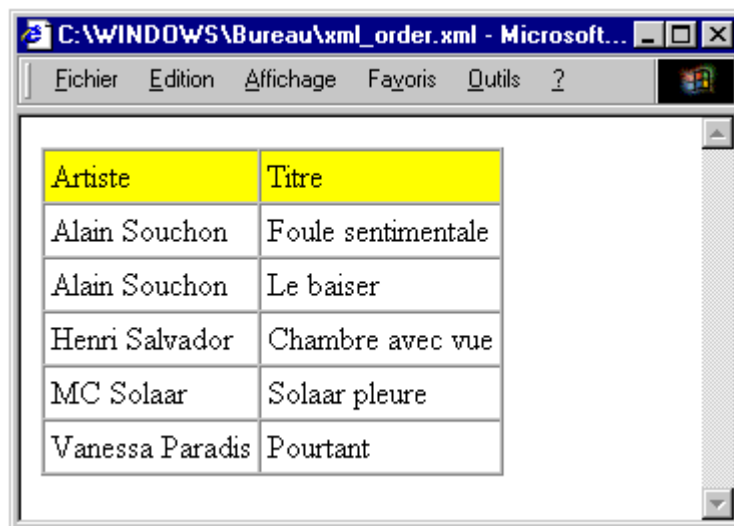
On revient au fichier XML et on y ajoute la balise pour y associer le fichier XSL.

```

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="xsl_order.xsl"?>
<compilation>
<mp3>
<titre>Foule sentimentale</titre>
<artiste>Alain Souchon</artiste>
</mp3>
etc...

```

Et miracle, notre stupide fichier XML plein de balises devient un tableau ordonné.



Artiste	Titre
Alain Souchon	Foule sentimentale
Alain Souchon	Le baiser
Henri Salvador	Chambre avec vue
MC Solaar	Solaar pleure
Vanessa Paradis	Pourtant

2.6. Filtrer avec le langage XSL

Filtrer avec le XSL

Le langage XSL permet aussi de filtrer les données du fichier XML associé selon des critères comme égal, pas égal, plus grand que, plus petit que.

Pour ce faire, il suffira d'utiliser l'attribut `select="chemin_d'accès[balise='xxx']"`.

Les opérateurs possibles sont :

= pour égal.

!= pour différent (non égal).

> pour plus grand que.

< pour plus petit que.

Un peu abstrait peut-être ? Rien de tel qu'un exemple...

Dans la compilation mp3, ne reprenons que les titres de l'artiste Alain Souchon. L'attribut select devient **select="'compilation/mp3[artiste='Alain Souchon']"**.

Elaboration du fichier

Reprenons notre fichier XML (inchangé).



```
<?xml version="1.0"?>
<compilation>
  <mp3>
    <titre>Foule sentimentale</titre>
    <artiste>Alain Souchon</artiste>
  </mp3>
  <mp3>
    <titre>Solaar pleure</titre>
    <artiste>MC Solaar</artiste>
  </mp3>
  <mp3>
    <titre>Le baiser</titre>
    <artiste>Alain Souchon</artiste>
  </mp3>
  <mp3>
    <titre>Pourtant</titre>
    <artiste>Vanessa Paradis</artiste>
  </mp3>
  <mp3>
    <titre>Chambre avec vue</titre>
    <artiste>Henri Salvador</artiste>
  </mp3>
</compilation>
```

Passons maintenant au fichier XSL



Nous allons reprendre dans notre compilation de mp3 en XML que les titres d'Alain Souchon.

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
  <html>
  <body>
```

```

<table border="1" cellspacing="0" cellpadding="3">
<tr bgcolor="#FFFF00">
<td>Titre</td>
<td>Artiste</td>
</tr>
<xsl:for-each select="'compilation/mp3[artiste='Alain Souchon']'">
<tr>
<td><xsl:value-of select="titre"/></td>
<td><xsl:value-of select="artiste"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

On enregistre le fichier sous le nom xsl_filter avec l'extension .xsl.

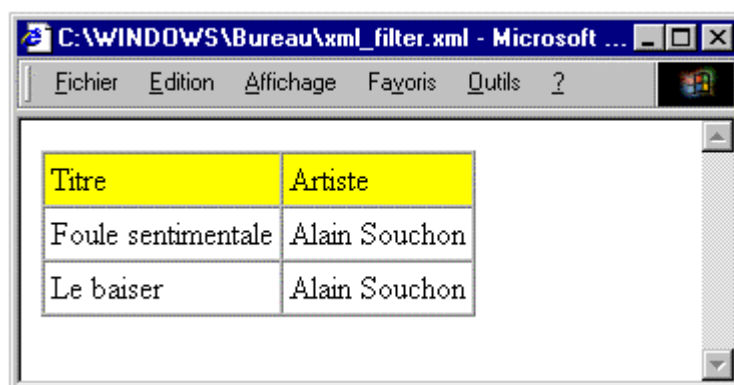
On revient au fichier XML et on y ajoute la balise pour y associer le fichier XSL.

```

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="xsl_filter.xsl"?>
<compilation>
<mp3>
<titre>Foule sentimentale</titre>
<artiste>Alain Souchon</artiste>
</mp3>
etc...

```

Et voilà notre fichier avec uniquement les titres d'Alain Souchon.



Titre	Artiste
Foule sentimentale	Alain Souchon
Le baiser	Alain Souchon

Trop facile, non ?

2.7. Choisir avec le XSL

Choisir avec le XSL

La balise `<xsl:if> ... </xsl:if>` permet d'effectuer un choix dans les données du fichier XML. On ajoutera l'attribut `match` où l'on indique l'élément choisi. Ce qui en résumé donne :

```
<xsl:if match=".[balise='xxx']">
  balises Html
</xsl:if>
```

Elaboration du fichier

Reprenons notre fichier XML (inchangé).



```
<?xml version="1.0"?>
<compilation>
  <mp3>
    <titre>Foule sentimentale</titre>
    <artiste>Alain Souchon</artiste>
  </mp3>
  <mp3>
    <titre>Solaar pleure</titre>
    <artiste>MC Solaar</artiste>
  </mp3>
  <mp3>
    <titre>Le baiser</titre>
    <artiste>Alain Souchon</artiste>
  </mp3>
  <mp3>
    <titre>Pourtant</titre>
    <artiste>Vanessa Paradis</artiste>
  </mp3>
  <mp3>
    <titre>Chambre avec vue</titre>
    <artiste>Henri Salvador</artiste>
  </mp3>
</compilation>
```

Passons maintenant au fichier XSL



Nous allons reprendre dans notre compilation de mp3 en XML que le(s) titre(s) de Vanessa Paradis.

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
  <html>
  <body>
    <table border="1" cellspacing="0" cellpadding="3">
      <tr bgcolor="#FFFF00">
        <td>Titre</td>
        <td>Artiste</td>
```

```

</tr>
<xsl:for-each select="compilation/mp3">
<xsl:if match=".[artiste='Vanessa Paradis']">
<tr>
<td><xsl:value-of select="titre"/></td>
<td><xsl:value-of select="artiste"/></td>
</tr>
</xsl:if>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

On enregistre le fichier sous le nom xsl_if avec l'extension .xsl.

On revient au fichier XML et on y ajoute la balise pour y associer le fichier XSL.

```

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="xsl_if.xsl"?>
<compilation>
<mp3>
<titre>Foule sentimentale</titre>
<artiste>Alain Souchon</artiste>
</mp3>
etc...

```

Et voilà notre fichier avec uniquement le titre de Vanessa Paradis.



2.8. Conditions et le XSL

Choix conditionnels et le XSL

Le XSL permet de faire un choix conditionnel par la balise `<xml:choose>`. A l'intérieur de cette balise, on peut déterminer une action lorsque une condition est vérifiée [`<xsl:when>`] et dans le cas contraire prévoir une autre action [`<xsl:otherwise>`].

```

<xsl:choose>
  condition vérifiée <xsl:when test=".[artiste='Alain Souchon']">
    <tr bgcolor="#00FF00">
    <td><xsl:value-of select="titre"/></td>

```

```

<td><xsl:value-of select="artiste"/></td>
</tr>
</xsl:when>

sinon
<xsl:otherwise>
<tr>
<td><xsl:value-of select="titre"/></td>
<td><xsl:value-of select="artiste"/></td>
</tr>
</xsl:otherwise>
</xsl:choose>

```

Un peu abstrait peut-être ? Rien de tel qu'un exemple...

Elaboration du fichier

Reprenons notre fichier XML (inchangé).



```

<?xml version="1.0"?>
<compilation>
<mp3>
<titre>Foule sentimentale</titre>
<artiste>Alain Souchon</artiste>
</mp3>
<mp3>
<titre>Solaar pleure</titre>
<artiste>MC Solaar</artiste>
</mp3>
<mp3>
<titre>Le baiser</titre>
<artiste>Alain Souchon</artiste>
</mp3>
<mp3>
<titre>Pourtant</titre>
<artiste>Vanessa Paradis</artiste>
</mp3>
<mp3>
<titre>Chambre avec vue</titre>
<artiste>Henri Salvador</artiste>
</mp3>
</compilation>

```

Passons maintenant au fichier XSL



Nous allons reprendre dans notre compilation de mp3 en XML tous les titres d'Alain Souchon que nous afficherons dans une colonne verte, les autres seront affichés normalement.

```

<?xml version='1.0'?>

```

```

<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<html>
<body>
<table border="1" cellspacing="0" cellpadding="3">
<tr bgcolor="#FFFF00">
<td>Artiste</td>
<td>Titre</td>
</tr>
<xsl:for-each select="compilation/mp3">
<xsl:choose>
<xsl:when test=".[artiste='Alain Souchon']">
<tr bgcolor="#00FF00">
<td><xsl:value-of select="titre"/></td>
<td><xsl:value-of select="artiste"/></td>
</tr>
</xsl:when>
<xsl:otherwise>
<tr>
<td><xsl:value-of select="titre"/></td>
<td><xsl:value-of select="artiste"/></td>
</tr>
</xsl:otherwise>
</xsl:choose>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

On enregistre le fichier sous le nom xsl_choose et l'extension .xsl.

On revient au fichier XML et on y ajoute la balise pour y associer le fichier XSL.

```

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="xsl_choose.xsl"?>
<compilation>
<mp3>
<titre>Foule sentimentale</titre>
<artiste>Alain Souchon</artiste>
</mp3>
etc...

```

Et voilà notre fichier avec uniquement les titres d'Alain Souchon.

Artiste	Titre
Foule sentimentale	Alain Souchon
Solaar pleure	MC Solaar
Le baiser	Alain Souchon
Pourtant	Vanessa Paradis
Chambre avec vue	Henri Salvador

2.9. Javascript et XSL

Une solution en Javascript

Nous avons expliqué dans les pages précédentes comment utiliser le XSL pour transformer un document XML sous forme de Html. La solution était d'ajouter une feuille de style XSL au fichier XML et de laisser faire le navigateur. Cependant, il n'est pas toujours souhaitable d'inclure une référence à une feuille de style dans un fichier XML car ce procédé ne fonctionnerait pas avec un navigateur qui ne reconnaît pas le XSL.

Une solution consiste à utiliser un script en Javascript pour faire la transformation du XML en Html. Cette transformation du XSL par le navigateur (soit côté client) est un des défis majeurs des navigateurs dans les années futures et devrait aboutir à des navigateurs spécialisés dans des tâches spécifiques comme le Braille, le Web parlant, les Pockets PC ou les téléphones mobiles.

Elaboration du fichier

Reprenons notre fichier XML [**xmldemo.xml**].



```
<?xml version="1.0"?>
<compilation>
<mp3>
<titre>Foule sentimentale</titre>
<artiste>Alain Souchon</artiste>
</mp3>
<mp3>
<titre>Solaar pleure</titre>
<artiste>MC Solaar</artiste>
</mp3>
<mp3>
<titre>Le baiser</titre>
<artiste>Alain Souchon</artiste>
</mp3>
<mp3>
<titre>Pourtant</titre>
```

```

<artiste>Vanessa Paradis</artiste>
</mp3>
<mp3>
<titre>Chambre avec vue</titre>
<artiste>Henri Salvador</artiste>
</mp3>
</compilation>

```

Passons maintenant au fichier XSL



On reprend simplement le fichier XSL du chapitre précédent [**xsl_choose.xsl**].

```

<?xml version='1.0'?>
<xsl:stylesheet
<xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<html>
<body>
<table border="1" cellspacing="0" cellpadding="3">
<tr bgcolor="#FFFF00">
<td>Artiste</td>
<td>Titre</td>
</tr>
<xsl:for-each select="compilation/mp3">
<xsl:choose>
<xsl:when test=".[artiste='Alain Souchon']">
<tr bgcolor="#00FF00">
<td><xsl:value-of select="titre"/></td>
<td><xsl:value-of select="artiste"/></td>
</tr>
</xsl:when>
<xsl:otherwise>
<tr>
<td><xsl:value-of select="titre"/></td>
<td><xsl:value-of select="artiste"/></td>
</tr>
</xsl:otherwise>
</xsl:choose>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

Construisons maintenant notre fichier Html avec son code Javascript qui va transformer nos fichiers XML et XSL en un seul fichier Html.

```

<html>
<body>
<script type="text/javascript">
// chargement du fichier XML
var xml = new ActiveXObject("Microsoft.XMLDOM")

```

```

xml.async = false
xml.load("xmldemo.xml")

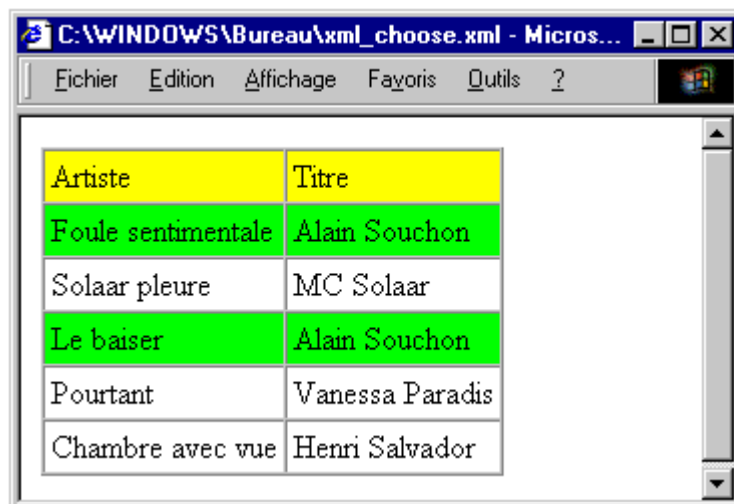
// chargement du fichier XSL
var xsl = new ActiveXObject("Microsoft.XMLDOM")
xsl.async = false
xsl.load("xsl_choose.xsl")

// transformation en Html
document.write(xml.transformNode(xsl))
</script>
</body>
</html>

```

On commence le script par un appel du XML Parser (XMLDOM) et le chargement en mémoire du fichier XML concerné. La seconde partie du script crée une autre instance du Parser et charge le document XSL en mémoire. La dernière ligne du code Javascript transforme le document XML selon le fichier XCL en écrit le résultat sous forme d'un document Html.

Simple, net et efficace ! On obtient (bien entendu) le même résultat qu'au chapitre précédent.



Artiste	Titre
Foule sentimentale	Alain Souchon
Solaar pleure	MC Solaar
Le baiser	Alain Souchon
Pourtant	Vanessa Paradis
Chambre avec vue	Henri Salvador

www.ccim.be/ccim328/xml/
 copyright © 2001

L'auteur

Van Lancker Luc
 Rue des Brasseurs, 22
 7700 Mouscron
 Belgium

Vanlancker.Luc@ccim.be