XML

Avertissement :
Une bonne partie de ce document (environ un quart) est composée d'extraits de sites Web et l'urgence de la rédaction ne m'a pas toujours permis de citer les sources.

INTRODUCTION

Pour tout savoir en quelques mots :

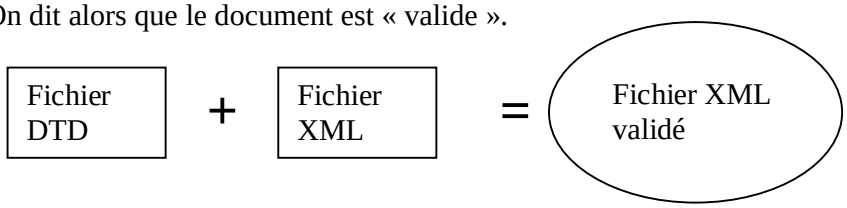
XML est un format de description qui se présente concrètement sous la forme dôun fichier texte possédant lôextension « .xml. ».
Il contient des balises dont les spécifications sont libres.
Par exemple :

```
<infos_produits >

	<produit>
		<nom>Confiture de myrtilles</nom>
		<prix_unitaire>12,50</prix_unitaire>
		<producteur>Herman</producteur>
		<infos_marketing>
			<particularites>Un peu de rhum</particularites>
			<gamme>2</gamme>
		</infos_marketing>
	</produit>

</infos_produits >
```

Pour que les balises puissent être utilisées par dôautres applications elles peuvent être décrites dans un autre fichier qui possède lôextension « .dtd ».

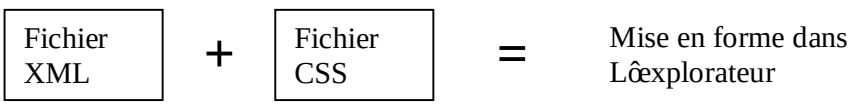
```
<!ELEMENT infos_produits (produit)>
<!ELEMENT produit (nom, prix_unitaire, producteur, infos_marketing)>
<!ELEMENT infos_marketing (particularites, gamme)>
```

On dit alors que le document est « valide ».



NB : Comme tout est toujours plus compliqué quôon veut bien le faire croire, les DTD sont amenées à être remplacées (mais peut-être pas avant longtemps !) par des fichiers XML-Shema.

Reste à afficher tout ça de façon agréable à lôaide de fichiers de description qui sôattacheront à préciser comment le contenu des balises va être présenté dans le browser.
On utilisera pour cela des fichier qui possèdent lôextension « .css ».



NB : Comme tout est toujours plus compliqué quôon veut bien le faire croire, les CSS sont amenées à être remplacées (mais peut-être pas avant longtemps !) par des fichiers XSL.

CHAPITRE I : Les bases

XML est un langage de description.
Description de tout ce quôon veux ; données, modes opératoires, règles, éléments matérielsé
XML (e**X**tensible **M**arkup **L**anguage = Langage de description extensible) se veut un outil de description universel.

Pourquoi ne pas continuer à utiliser HTML ?
Parce que HTML est dôabord un outil de présentation. Au fur et à mesure de ses évolutions, il sôest enrichi (dynamisme, intégration de scripts) mais sa syntaxe est indépendante de son contenu.
Avec XML, les balises prennent du **SENS** ; l'utilisateur a la possibilité d'inventer de nouveaux tags afin de mieux décrire le contenu de sa page. Il a ainsi la possibilité de mieux structurer les informations de sa page web avec la possibilité de mettre en avant des mots ou paragraphes par rapport à du simple texte.
De plus, les informations importantes mises en évidence de cette manière par les marqueurs (=balise =tag) permettraient bien d'autres utilisations de l'information. Par exemple, avec un bout de code qui analyserait le contenu de la page, on pourrait facilement alimenter une base de données.

Un peu de concret, voici un source XML :

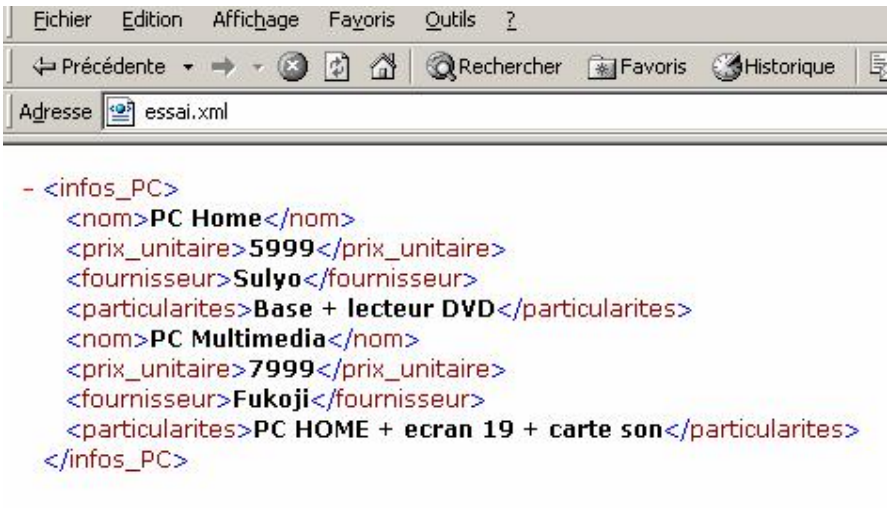
```
<infos_PC>
    <nom>PC Home</nom>
    <prix_unitaire>5999</prix_unitaire>
    <fournisseur>Sulyo</fournisseur>
    <particularites>Base + lecteur DVD</particularites>
    <nom>PC Multimedia</nom>
    <prix_unitaire>7999</prix_unitaire>
    <fournisseur>Fukoji</fournisseur>
    <particularites>PC HOME + ecran 19 + carte son</particularites>
 </ infos_PC>
```

(fichier 001_intro.xml)

Celui-ci décrit succinctement des modèles dôordinateur.
On peut remarquer que toutes les balises marchent par paire, que la deuxième balise associée contient le caractère « / » et quôil existe une balise « racine » (<*infos_PC*>) qui encadre toutes les autres.
Si nous voulons récupérer la liste de fournisseurs, il suffit dôanalyser le document et mettre de coté ce qui se trouve entre les balises < *fournisseur* >.
Avec un peu de chance, on utilisera un outil qui fait tout le travail pour nousé

Pour voir le résultat du source XML (<*infos_PC*>***é***) il suffit de le taper dans un éditeur de texte quelconque et de le sauvegarder au format texte avec lôextension « .xml » .

Essayez de l'ouvrir avec IE, vous devriez obtenir ceci :



Le résultat est plutôt décevant.
C'est donc ça XML ?
Pas vraiment : en réalité XML regroupe tout un ensemble de techniques qui permettent de structurer des informations, contrôler la cohérence de celles-ci et les mettre en forme.
Au fait, saviez-vous que XML était un sous-ensemble de **SGML** ?
SGML est une sorte de langage pour écrire des langages.
Mais celui-ci est assez compliqué à mettre en î uvreé

***

Passons aux choses sérieuses :
Dans l'exemple précédent nous avons vu que la balise « <fournisseur> » pouvait nous permettre, via un outil que nous ne connaissons pas encore, de récupérer la liste des fournisseurs. Encore faut-il que nous connaissions son nom (<fournisseur>).
Si un autre site propose des informations de même type, mais que la balise contenant le nom de son fournisseur s'appelle « <fournisseurPC> » notre recherche échouera.
Même pour nous, dans la création des pages XML, il serait bon qu'une fois pour toutes nous utilisions les mêmes balises pour décrire les mêmes données.
Pour ce faire nous pouvons imaginer un document de référence qui décrirait l'ensemble des balises que nous serions en droit d'utiliser.
Ce document, nous pourrions le communiquer à toutes les personnes susceptibles d'exploiter les mêmes structures.
Dans l'idéal, ce document serait centralisé par un organisme connu de tous, et qui le mettrait à disposition du plus grand nombre.
Allons plus loin ; ce document pourrait même être interprété par le navigateur afin qu'il vérifie que le source XML s'inscrit dans le cadre de ses spécifications.
« And voilà » la fameuse **DTD** (**D**ocument **T**ype **D**efinition) !!

Notre source XML bien formé et validé, il ne reste plus quôà le mettre en forme.
Rappelons le, XML est un langage de description de données, pas de mise en page.
Chaque chose à sa place !
Là encore nous pouvons imaginer un autre document qui préciserait, pour chaque balise, comment elle doit être affichée. Pour éviter de réinventer la roue, les feuilles de style utilisées pour HTML sont utilisées : **CSS** (Cascading Style Sheets).
Lôinconvénient des CSS est quôelles ont été conçues pour HTML et sont donc non conformes aux principes du XML. Côest pourquoi a été créé le **XSL** (eXtensible Style Language) qui permet aussi de mettre en page les sources XML.
Ca y est, on nage dans les siglesé et on ne sait toujours pas faire un affichage sympa en XML.
Dans toutes les documentations existantes, vous trouverez de longues explications ayant pour but de vous expliquer que XML est simple et révolutionnaire, mais si les principes de base ont lôambition dôêtre clair, les outils qui gravitent autour sôaffranchissent allègrement de cette noble idéeé

***

Essayons dôy voir un peu plus clair en reprenant le premier source :

```
<infos_PC>
    <nom>PC Home</nom>
    <prix_unitaire>5999</prix_unitaire>
    <fournisseur>Sulyo</fournisseur>
    <particularites>Base + lecteur DVD</particularites>
    <nom>PC Multimedia</nom>
    <prix_unitaire>7999</prix_unitaire>
    <fournisseur>Fukoji</fournisseur>
    <particularites>PC HOME + ecran 19 + carte son</particularites>
  </ infos_PC>
```

Rendons le **valide** en le faisant dépendre dôune DTD.
Celle-ci sôattachera à décrire le *vocabulaire* utilisé dans le source XML.
Il existe deux façons dôutiliser une DTD

☞1 ï En la décrivant au sein même du source XML

☞2 ï En la décrivant dans un autre fichier, dans ce cas il faudra indiquer dans le source XML où la trouver (ce peut être sur un site http).

☞1 ï Dans le source XML, une DTD se déclare selon la syntaxe suivante :

```
< !DOCTYPE élément_racine [
        < !ELEMENT nom_element (règles_dtd)>
        < !ELEMENT nom_element (règles_dtd)>

        < !ELEMENT nom_element (règles_dtd)>
        é
        é
        ]>
```

« élément_racine » correspond à la racine du document (c.a.d la première balise ; qui encadre toutes les autres)
« [règles_dtd]> » permet dôapporter des précisons structurelles sur le contenu ; pour lôinstant nous mettrons le nom dôautres éléments contenus ou « #PCDATA » pour indiquer que lô élément contient du texte.

Le nouveau fichier doit ressembler à ça :

```
<!DOCTYPE infos_PC [
        <!ELEMENT infos_PC (nom,prix_unitaire,fournisseur,particularites)>
        <!ELEMENT nom (#PCDATA)>
        <!ELEMENT prix_unitaire (#PCDATA)>
```

```
        <!ELEMENT fournisseur (#PCDATA)>
        <!ELEMENT particularites (#PCDATA)>
                        ]>

<infos_PC>
        <nom>PC Home</nom>
        <prix_unitaire>5999</prix_unitaire>
        <fournisseur>Sulyo</fournisseur>
        <particularites>Base + lecteur DVD</particularites>
        <nom>PC Multimedia</nom>
        <prix_unitaire>7999</prix_unitaire>
        <fournisseur>Fukoji</fournisseur>
        <particularites>PC HOME + ecran 19 + carte son</particularites>
 </infos_PC>
```

(fichier 002_intro.xml)

***

Il est possible de visualiser le document XML dans internet explorer, celui-ci lôaffichera si il est syntaxiquement correct ; on dit alors quôil est ***bien formé***.
Mais comment vérifier quôil est ***valide*** ? Cêst à dire que son contenu correspond à sa DTD ?
En attendant que le navigateur prenne en charge cette fonctionnalité (prévue pour IE6) il faut utiliser des outils disponibles en téléchargement.
(Au début je nôai pas trouvé dôoutil satisfaisant, aussi ai-je développé un programme VB qui utilise la Dll msxml.dll, qui permet de valider et parcourir un modèle objet contenant le document xml. *Le programme avec source est disponible*). Un outil pratique est « Architag Real-time XML Editor » pris sur http://architag.com/editor à condition que la DTD soit dans le document XML (Mais la dernière fois que je suis retourné sur ce site, il nôy était plusé .).
Le principe de cet outil est de valider au fil de lôeau le document XML que vous tapez dans une zone de texte.
Si le source ne vous est pas livré avec cette documentation, vous pourrez le trouvez en annexe → Valideur Temps Reel.

Sur le source xml précité, la validation échoue :
Impossible de charger le document à cause des erreurs suivantes.
Erreur numéro : -1072898028
Libellé : Contenu d'élément non valide selon la DTD/schéma.
Ligne numéro : 17
En position : 7
--------> : <nom>PC Multimedia</nom>

Lôexplication est que le bloc « infos_PC » contient les informations pour plusieurs machines, sans séparation.
Quand peut-on savoir quôon passe à la description dôune autre machine ?
Pour cela, il est nécessaire de modifier un peu le contenu de notre document comme suit :

```
<!DOCTYPE infos_PC [
        <!ELEMENT infos_PC (modele*)>
        <!ELEMENT modele (nom,prix_unitaire,fournisseur,particularites)>
        <!ELEMENT nom (#PCDATA)>
        <!ELEMENT prix_unitaire (#PCDATA)>
        <!ELEMENT fournisseur (#PCDATA)>
        <!ELEMENT particularites (#PCDATA)>
                ]>

<infos_PC>
        <modele>
                <nom>PC Home</nom>
                <prix_unitaire>5999</prix_unitaire>
                <fournisseur>Sulyo</fournisseur>
                <particularites>Base + lecteur DVD</particularites>
        </modele>
        <modele>
                <nom>PC Multimedia</nom>
                <prix_unitaire>7999</prix_unitaire>
                <fournisseur>Fukoji</fournisseur>
                <particularites>PC HOME + ecran 19 + carte son</particularites>
        </modele>

</infos_PC>
```

← Coder et tester ce source

(fichier 003_intro.xml)

☞2 ï Enregistrons maintenant la DTD dans un autre fichier :

```
<!ELEMENT infos_PC (modele*)>
<!ELEMENT modele (nom,prix_unitaire,fournisseur,particularites)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prix_unitaire (#PCDATA)>
<!ELEMENT fournisseur (#PCDATA)>
<!ELEMENT particularites (#PCDATA)>
```

(fichier 004_intro.**dtd**)

Et déclarons la DTD externe via lôinstruction SYSTEM :

```
<!DOCTYPE infos_PC SYSTEM "004_intro.dtd">

<infos_PC>
        <modele>
        <nom>PC Home</nom>
        <prix_unitaire>5999</prix_unitaire>
        <fournisseur>Sulyo</fournisseur>
        <particularites>Base + lecteur DVD</particularites>
        </modele>
        <modele>
        <nom>PC Multimedia</nom>
        <prix_unitaire>7999</prix_unitaire>
        <fournisseur>Fukoji</fournisseur>
        <particularites>PC HOME + ecran 19 + carte son</particularites>
        </modele>
</infos_PC>
```

(fichier 004_intro.xml)

 Exercice :

Faire différents changements dans la DTD externe et/ou dans le source XML (Mettre une lettre en majuscule, rajouter un élément etcé) afin dôobserver les problèmes de fonctionnements.

Exercice 1 :

Coder et tester la dtd et le source XML suivants :

Un commentaire commence par
< !--
et finis par -->

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!-- dtd dialogue -->

<!ELEMENT dialogue (situation?, replique+) >
  <!-- un dialogue est composé de 0 ou 1 situation puis de au moins 1 replique -->
<!ELEMENT situation (#PCDATA) >
    <!--  une situation est une donnée textuelle PCDATA (Parsed Character Data) -->
<!ELEMENT replique (personnage, texte) >
    <!--  une replique est composée de 1 personnage puis de 1 texte-->
<!ELEMENT personnage (#PCDATA) >
<!ELEMENT texte (#PCDATA) >
```

(fichier 005_intro.**dtd**)

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!-- Encoding à mettre à la fois dans la DTD et dans le docu xml ... -->

<!DOCTYPE dialogue SYSTEM "005_intro.dtd">

<!-- extrait de Tartuffe de Molière -->
<dialogue>
<situation>acte I, Scene 1 : madame pernelle et flipote sa servante, elmire, mariane, dorine, damis, cléante.
</situation>
<replique>
   <personnage>madame pernelle</personnage>
   <texte>Allons, Flipote, allons, que d'eux je me délivre.</texte>
</replique>
<replique>
   <personnage>elmire</personnage>
   <texte>Vous marchez d'un tel pas qu'on a peine à vous suivre.</texte>
</replique>
</dialogue>
```

(fichier 005_intro.xml)

Quelques précisions...

On a la possibilité de "décorer" les types d'éléments par des opérateurs de cardinalité :

nom **?**	Permet de définir un type d'élément optionnel. Le "**?**" peut porter sur un seul type d'élément ou sur un ensemble de type d'éléments regroupé entre parenthèses. Ex : <!ELEMENT personne(nom, prénom, téléphone, portable**?**)>, définit un type d'élément personne contenant les types d'éléments nom, prénom, téléphone et de manière optionnelle portable, dans cet ordre.
nom *****	Permet de définir un type d'élément pouvant apparaître aucune ou plusieurs fois. Le "*****" peut porter sur un seul type d'élément ou sur un ensemble de type d'éléments regroupé entre parenthèses. Ex: <!ELEMENT personne(nom, prénom, téléphone*****)>, définit un type d'élément personne contenant les types d'éléments nom, prénom, et zéro ou plusieurs fois le type d'élément téléphone.
nom **+**	Permet de définir un type d'élément pouvant apparaître une ou plusieurs fois. Le "**+**" peut porter sur un seul type d'élément ou sur un ensemble de type d'éléments regroupé entre parenthèses. Ex: <!ELEMENT personne(nom, prénom, téléphone**+**)>, définit un type d'élément personne contenant les types d'éléments nom, prénom, et une ou plusieurs fois le type d'élément téléphone.

☞ **Pour en savoir plus, consulter lôannexe « Contenu dôune DTD »**

Exercice 2 :

Etablir la DTD du document suivant, sachant que lôélément fleuriste est optionnel et quôil contient nécessairement soit un élément ville soit un élément lieu.

```
<composition>
<auteur>Floristore</auteur>
<date_de_creation>29/02/2001</date_de_creation>
<fleuriste>
        <ville>Paris</ville>
</fleuriste>
<composants>
  <vase>
    <matiere>Porcelaine</matiere>
    <taille>50x70 cm</taille>
  </vase>
  <fleurs>
    <nature>Europeenne</nature>
    <item>Tulipe</item>
    <item>glaieul</item>
    <item>Marguerite</item>
  </fleurs>
</composants>
</composition>
```

Exercice 3 :

Créer le plus petit document xml possible à partir de la dtd suivante

```
<!DOCTYPE aa [
<!ELEMENT aa (aa1,aa2?)>
 <!ELEMENT aa1 (#PCDATA)>
 <!ELEMENT aa2 (#PCDATA)>
]>
```

Exercice 4 :

Etablir la DTD et le document XML qui permettent dôavoir un élément racine aa contenant un élément aa1 suivi soit dôun élément aa2, soit de un ou plusieurs éléments aa3.

 Exercice 5 :

Créer la DTD correspondant au source XML suivant

```
<?xml version="1.0"?>
<!DOCTYPE compactdiscs SYSTEM "cds.dtd">
<compactdiscs>
  <compactdisc>
    <artist type="individuel">Charles Aznavour</artist>
    <title numberoftracks="4">Je môvoyais deja</title>
    <tracks>
      <track> Je môvoyais deja </track>
      <track>Les plaisirs demodes</track>
      <track>Les comediens</track>
      <track>La mamma</track>
    </tracks>
    <price>189 FF</price>
    </compactdisc>
    <compactdisc>
    <artist type="groupe">Les beatles</artist>
    <title numberoftracks="5">Yellow submarine</title>
    <tracks>
    <track>Yellow submarine </track>
    <track>Michele</track>
    <track>Let it be</track>
    <track>Just you need</track>
    <track>Hard dayôs night</track>
    </tracks>
    <price>79 FF</price>
  </compactdisc>
</compactdiscs>
```

RAPPELS ET PRECISIONS :

Soit le source XML suivant

```
L1 -  <?xml version="1.0" encoding= "ISO-8859-1" ?>
L2 -  <!DOCTYPE infos_PC
L3 -        [
L4 -        <!ELEMENT infos_PC (modele+)>
L5 -        <!ELEMENT modele (nom,prix_unitaire,fournisseur,particularites*)>
L6 -        <!ELEMENT nom (#PCDATA)>
L7 -        <!ELEMENT prix_unitaire (#PCDATA)>
L8 -        <!ELEMENT fournisseur (#PCDATA)>
L9 -        <!ELEMENT particularites (#PCDATA)>
L10 -       ]>
L11 -
L12 - <infos_PC>
L13 - <modele>
L14 -       <nom>PC Home</nom>
L15 -       <prix_unitaire>5999</prix_unitaire>
L16 -       <fournisseur>Sulyo</fournisseur>
L17 - </modele>
L18 - <modele>
L19 -       <nom>PC Multimedia</nom>
L20 -       <prix_unitaire>7999</prix_unitaire>
L21 -       <fournisseur>Fukoji</fournisseur>
L22 -       <particularites>PC HOME + ecran 19 + carte son</particularites>
L23 - </modele>
L24 - </infos_PC>
```

L1 - La première ligne s'appelle la déclaration. Elle commence par « < ? » et se termine par « ?>.
Elle décrit la version de XML et le codage.
La version 1.0 est la seule à l'heure actuelle. Le codage indique le type de caractère obtenu à l'appui d'une touche du clavier. La norme « ISO-8859-1 » rend possible l'affichage de tous les caractères de la langue de Molière.
Censée être obligatoire, on a pu s'en passer dans les exemples précédents.
Cependant, dès l'utilisation de caractères accentués, elle sera indispensable.

L2 - Début de déclaration de la DTD.
Un document est dit « **valide** » si il est bien formé syntaxiquement et si il respecte la structure définie par la DTD. Celle-ci peut être externe, on utilise alors l'instruction « SYSTEM ».
« <!DOCTYPE infos_PC » Précise le nom de l'**élément racine** du document.

L3 - Encadre la description des **élements**.

L4 - L'élément « infos_pc » doit contenir un élément « modele »
Le signe « + » signifie qu'il doit y en avoir au moins un.
NB :
Sans signe particulier, un et un seul élément doit être présent.
Avec le signe « * » zéro ou plusieurs éléments sont acceptés.

L5 - L'élément « *modele* » doit contenir une et une seule fois les éléments
- *« nom »*
- *« prix_unitaire »*
- *« fournisseur »*

et zéro ou plusieurs éléments « *particularites* ».

L6 - Lôélément « *nom* » est destiné à contenir du texte.
→ Voir « Contenu dôune DTD » en Annexe ←
L7 - Lôélément « *prix_unitaire* » est destiné à contenir du texte.
L8 - Lôélément « *fournisseur* » est destiné à contenir du texte.
L9 - Lôélément « *particularites* » est destiné à contenir du texte.
L10 - Balise de fermeture de déclaration de DTD. NB : Dans une dtd externe ces balises nôapparaissent pas.
L11 -
L12 - Elément racine.
L13 - Elément fils de lôélément racine.
L14 - Elément fils de lôélément modèle (voir le D.O.M en Annexe).
é
é
é
L24 - Fermeture de lôélément racine et donc fin du document.

CHAPITRE II : La présentation

Il existe plusieurs moyens de contrôler la présentation dôun document XML.
Parmi ceux-ci, le plus utilisé actuellement est CSS : Cascading Style Sheets.
Les spécifications sont sur http://www.w3.org/TR/REC-CSS1

Depuis les versions 4.0 de Navigator, dôOpera et dôInternet Explorer, il est devenu possible de tirer profit des spécifications du consortium W3C concernant les feuilles de style. Cet outil, souvent complémentaire du HTML, procure une totale maîtrise de lôaffichage des pages web. Il existe aujourdôhui deux niveaux de spécifications du langage (CSS1 et CSS2), le second niveau nôétant implémenté que par les navigateurs les plus récents.

Une feuille de style définit un modèle pour une ou plusieurs pages contenant des balises.
Un style associé à une balise en particulier ne sera précisé quôune seule fois, simplifiant considérablement les changements ultérieurs et réduisant sensiblement le poids des pages.
Par exemple, on pourra regrouper, au sein dôun unique fichier (.css), toutes les règles de style qui définissent lôaspect visuel dôun site entier. Cela sôavère particulièrement pratique pour des sites professionnels dont la charte graphique est rigoureuse, et plus encore quand celle-ci doit subir un renouvellement en profondeur.
A quoi ressemble une règle CSS?
Prenons lôexemple suivant : imaginons que notre document XML utilise deux types de police (disons : Arial 12pt en italique rouge et Verdana 10pt en bleu) ; lôune appliquée aux références produits et lôautre au prix. Cela nécessiterait, si le document XML était traduit vers le HTML (pour la présentation), dôutiliser la balise <FONT> et sa cohorte dôattributs pour chaque changement de police. C'est à la fois fastidieux et préjudiciable au temps de chargement de la page. Mais en utilisant les feuilles de style, ce problème disparaît : il nous suffit de caractériser chacune de nos polices avec les balises correspondantes du document (<RefProduit> et <Prix> par exemple), balises auxquelles on aura préalablement associé le style que nous souhaitons :

RefProduit {font-family:Arial; font-size:12pt; font-weight:italic; color:red;}

et:

Prix {font-family:Verdana; font-size:10pt; color:blue;}

Le modèle étant le suivant:

Comment combiner CSS et XML (ou HTML)?

En HTML il existe plusieurs façons distinctes dôinsérer ces définitions de style,
les *inline style sheets* ou, en français, les feuilles de style locales,
les *embedded style sheets* ou feuilles de style internes,
les *external style sheets* ou feuilles de style externes.

Les inline style sheets ont l'avantage de pouvoir être insérées partout dans notre code.
Si nous n'avons que quelques mots ou quelques lignes à modifier, elles peuvent s'avérer bien pratiques.

Les embedded style sheets, par contre, sont situées dans la partie head d'une page et exercent leur "pouvoir" sur la page entière. Dans certaines situations, elles sont donc plus efficaces que les feuilles de style locales et nous offrent, de toute manière, la possibilité de gérer plus de code en même temps.

Par exemple :

```
<HEAD>
  <TITLE> ... </TITLE>
  <STYLE type="text/css">
    BODY {color: darkred;}
    H1 {color: darkblue;}
  </STYLE>
</HEAD>

<BODY bgcolor="#ffffff">
  <P>Texte affiché en rouge foncé.</P>
  <H1>Titre affiché en bleu foncé.</H1>
  <DIV>Texte affiché en rouge foncé.</DIV>
  ...
</BODY>
```

Tester ce code en lôenregistrant au format html

De même quôil est possible de combiner plusieurs règles de styles, la priorité sôexerçant comme suit :

1. marqueurs et attributs HTML
2. commandes des feuilles de style locales
3. commandes des feuilles de style internes
4. commandes des feuilles de style externes

Donc du code le plus près des balises vers lôexterieur.

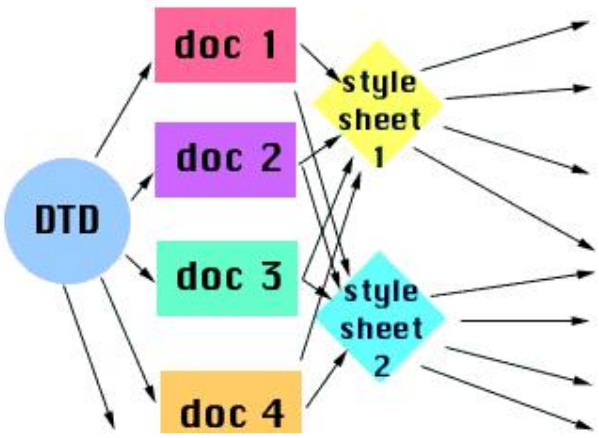
Mais la plupart du temps on préfère établir un lien vers le fichier .css qui contient toutes les règles (*external style sheets*). Cela impose de créer un fichier "*mamisenforme1.css"* par exemple, qui contiendrait les lignes suivantes :

```
RefProduit {font-family:Arial; font-size:14pt; font-weight:italic; color:red}
Prix {font-family:Verdana; font-size:10pt; color:blue}
```

Puis, dans la partie déclarative du document de notre fichier XML on écrira :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="mamisenforme1.css"?>
<Article>
  <RefProduit>PC Multimedia  </RefProduit>
  <Prix>1981 Euros</Prix>
</Article>
```

Ainsi, plusieurs liens externes peuvent être « importés » dans le document, et dans le cas de règles conflictuelles, la dernière définition lôemporte.
Lorsqu'un appel à plusieurs feuilles de style externe est pratiqué, le résultat est normalement une combinaison en cascade d'ensembles de définitions de style.
C'est-à-dire, les styles spécifiés pour des balises HTML seront tous chargés à condition qu'ils soient tous affectés à des éléments différents. Dans le cas contraire, les doublons seront écrasés en sachant que les derniers styles énoncés ont la priorité.



Voici un joli exemple qui permet de changer le look de la barre dôascenseur (fonctionne à partir de IE5.5) :

```
<head>
<style type="text/css"><!--

body {
 scrollbar-face-color: #FF9900;
 scrollbar-shadow-color: #FFCC00;
 scrollbar-highlight-color: #FFFFFF;
 scrollbar-3dlight-color: FF6600;
 scrollbar-darkshadow-color: #993300;
 scrollbar-track-color: #6699CC;
 scrollbar-arrow-color: #FFFFCC;
}
//--></style>

</head>

<body>
toto<br>
toto<br>
toto<br>
</body>
```

scrollbar-face-color: *couleur du dessus des boutons et de la barre de défilement.*
scrollbar-shadow-color: *couleur sombre du relief des boutons (noir conseillé).*
scrollbar-highlight-color: *couleur claire du relief des boutons (blanc conseillé).*
scrollbar-3dlight-color: *couleur claire du relief des boutons quand ceux-ci ne sont pas enfoncés (noir conseillé).*
scrollbar-darkshadow-color: *couleur sombre du relief des boutons quand ceux-ci ne sont pas enfoncés (noir conseillé).*
scrollbar-track-color*: couleur du fond de la barre.*
scrollbar-arrow-color : *Couleur des flèches de navigation.*

Référence des attributs affectés aux styles (http://www.brown.edu/webmaster/webpublishing/tutorials/CSS/)non exhaustive.

Text styles:
color (hex, RGB, or keyword - see background-color below)
font-weight (normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900 - each value is one degree bolder than the previous, normal is equivalent to 400, bold is equivalent to 700)
font-family (serif, sans-serif, cursive, fantasy, monospace)
font-size (xx-small, x-small, small, medium, large, x-large, xx-large, or relative values in px pt in, or other)
font-style (normal, italic, oblique)
text-decoration (underline, overline, line-through, blink)
text-transform (capitalize, uppercase, lowercase)
letter-spacing (percentages or absolute values in px, pt, in, or other)
word-spacing (same as above)
line-height (same as above)
text-indent (same as above)
text-align (left, right, center, justify)
vertical-align (baseline, middle, sub, super, text-top, text-bottom, top, bottom, or a percentage)
Background styles:
background-color (hex, RGB, or keyword - keywords are: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow; activeborder, activecaption, appworkspace, background, buttonface, buttonhighlight, buttontext, captiontext, graytext, highlight, highlighttext, inactiveborder Inactive window border, inactivecaption, inactivecaptiontext, infobackground, infotext, menu, menutext, scrollbar, threeddarkshadow, threedface, threedhighlight, threedlightshadow, threedshadow, window, windowframe, windowtext)
background-image (URI)
background-attachment (scroll, fixed)
background-repeat (repeat, repeat-x, repeat-y, no-repeat)
background-position (top, left, center, right, bottom [alone or in combination, separated by spaces], percentages, or values in px, pt, in, or other)
Border styles:
border-top-width (thin, medium, thick, or values in px, pt, in, or other)
border-right-width (thin, medium, thick, or values in px, pt, in, or other)
border-bottom-width (thin, medium, thick, or values in px, pt, in, or other)
border-left-width (thin, medium, thick, or values in px, pt, in, or other)
border-width (thin, medium, thick, or values in px, pt, in, or other)
border-color (hex, RGB, or keyword - see background-color above)
border-style (none, dotted, dashed, solid, double, groove, ridge, inset, outset)
border-top (combination of values above, separated by spaces)
border-right (combination of values above, separated by spaces)
border-bottom (combination of values above, separated by spaces)
border-left (combination of values above, separated by spaces)
border (combination of values above, separated by spaces)
Margin styles:
margin-top (percentages or values in px, pt, in, or other)
margin-left (percentages or values in px, pt, in, or other)
margin-bottom (percentages or values in px, pt, in, or other)
margin-right (percentages or values in px, pt, in, or other)
margin (percentages or values in px, pt, in, or other)
Padding styles:
padding-top (percentages or values in px, pt, in, or other)
padding-left (percentages or values in px, pt, in, or other)
padding-bottom (percentages or values in px, pt, in, or other)
padding-right (percentages or values in px, pt, in, or other)
padding (percentages or values in px, pt, in, or other)

Element type styles:
display (none, block, inline, list-item, table, inline-table, table-column, table-header-group, table-footer-group, table-row, table-cell, table-caption)
white-space (normal, pre, nowrap)
list-style-type (none, disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha)
list-style-image (URI)
list-style-position (inside, outside)
list-style (combination of values above, separated by spaces)
row-span (integer)
column-span (integer)
border-collapse (collapse, separate)
table-layout (auto, fixed)
User interface:
cursor (URI, or auto, crosshair, default, pointer, move, text, wait, help, *-resize [* is replaced with N, S, E, W, NE, NW, SE, SW for each of the edges (north, south, east, west and so on)].
outline-color (invert, or hex, RGB, or keyword - see background-color above)
outline-style (none, dotted, dashed, solid, double, groove, ridge, inset, outset)
outline-width (thin, medium, thick, or values in px, pt, in, or other)
outline (combination of values above, separated by spaces)
Layout styles:
position
top
left
bottom
right
width
min-width
max-width
height
z-index
visibility
overflow
float
clear
clip

On peut également en trouver une liste dans le MSDN :
SDK Platform/Services Internet, Intranet et Extranet/Dynamic HTML/DHTML References/CSS Attributes
Ou dans IE taper comme adresse :
mk:@MSITStore:C:\Program%20Files\Microsoft%20Visual%20Studio\MSDN98\98VSa\1036\IERef.chm::/inet401/help/dhtml/references/css/attributes.htm#ie40_cssref

Editeur de feuilles de style

Voici une liste fournie par allhtml.com (http://allhtml.com/telechargement/categorie2.php)

Nom	Nb Tél.	OS	Licence	Taille	Note ALL HTML
Style Master CSS	57	Pc Mac	Shareware	2.77 Mo	★★★★☆
TopStyle	586	Pc	Freeware	1.33 Mo	★★★★★
CoffeeCup StyleSheet	7	Pc	Shareware	1.51 Mo	★★★☆☆
Sheet Stylist	9	Pc	Shareware	1.46 Mo	★★★☆☆
Cascade	72	Pc	Freeware	1.38 Mo	★★★★☆

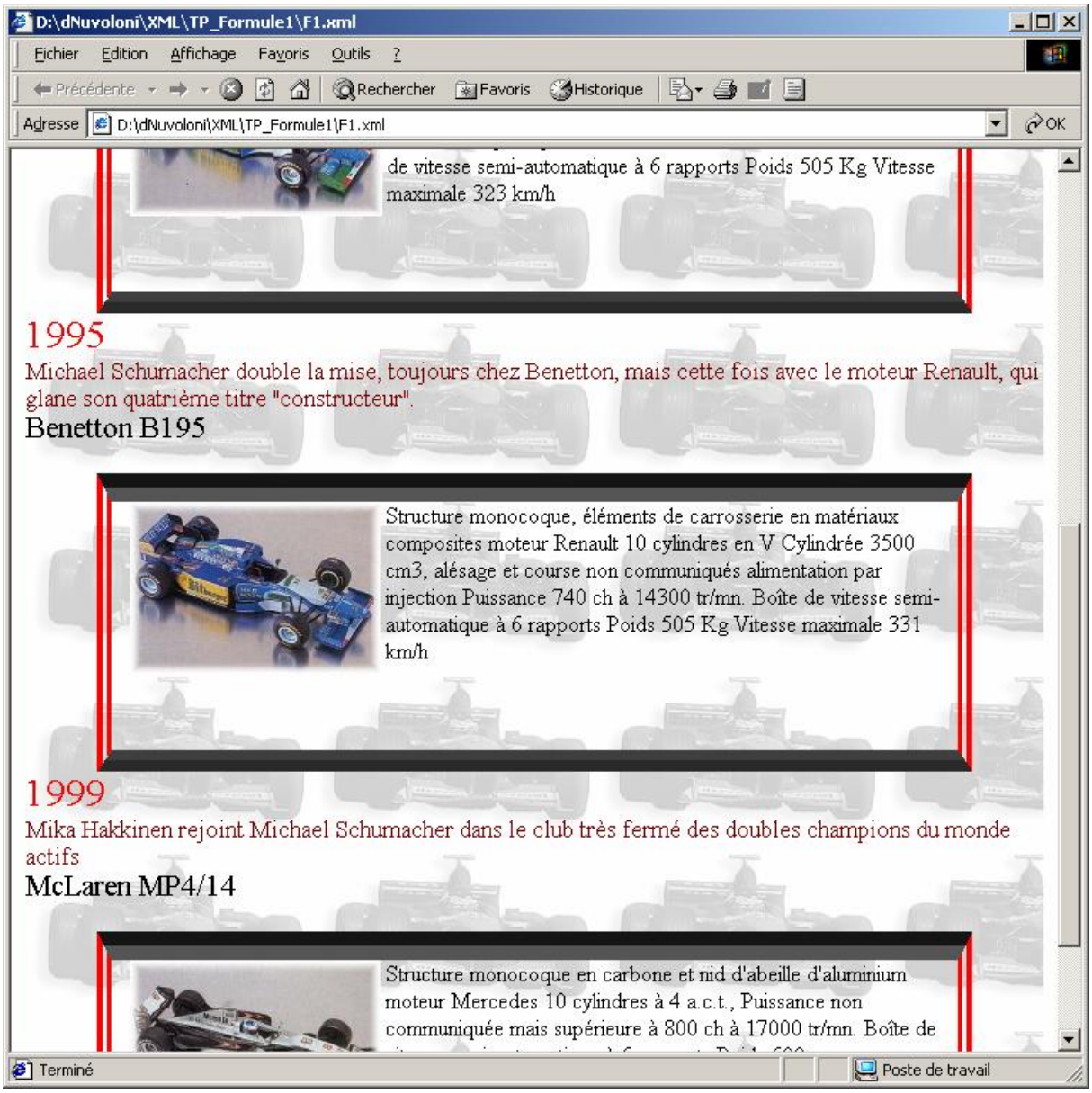
☞ **TopStyle Pro 2.5** a la meilleure note et en plus il est en libre évaluation sur http://www.bradsoft.com/topstyle/index.asp

TP « Formule 1 »

A partir du texte suivant :

1990
Ayrton Senna sur McLaren Honda, bat Alain Prost, pourtant passé chez Ferrari
McLaren MP4/5B
Structure monocoque, éléments de carrosserie en matériaux composites moteur Honda 10 cylindres en V à 72° Cylindrée 3498 cm3, alésage 92 mm course 52,5 mm alimentation par injection Puissance 690 ch à 13000 tr/mn. Boîte de vitesse à 6 rapports Poids 500 Kg Vitesse maximale 325 km/h
1994
Michael Schumacher offre à Ford le titre "pilote", mais c'est Renault, malgré le décès de Ayrton Senna, qui coiffe la couronne "constructeur".
Benetton B194
Structure monocoque, éléments de carrosserie en matériaux composites moteur Ford Zetec à 8 cylindres en V à 75°, cylindrée 3494 cm3, alésage et course non communiqués, alimentation par injection Puissance 730 ch à 14500 tr/mn. Boîte de vitesse semi-automatique à 6 rapports Poids 505 Kg Vitesse maximale 323 km/h
1995
Michael Schumacher double la mise, toujours chez Benetton, mais cette fois avec le moteur Renault, qui glane son quatrième titre "constructeur".
Benetton B195
Structure monocoque, éléments de carrosserie en matériaux composites moteur Renault 10 cylindres en V Cylindrée 3500 cm3, alésage et course non communiqués alimentation par injection Puissance 740 ch à 14300 tr/mn. Boîte de vitesse semi-automatique à 6 rapports Poids 505 Kg Vitesse maximale 331 km/h
1999
Mika Hakkinen rejoint Michael Schumacher dans le club très fermé des doubles champions du monde actifs
McLaren MP4/14
Structure monocoque en carbone et nid d'abeille d'aluminium moteur Mercedes 10 cylindres à 4 a.c.t., Puissance non communiquée mais supérieure à 800 ch à 17000 tr/mn. Boîte de vitesse semi-automatique à 6 rapports Poids 600 Kg

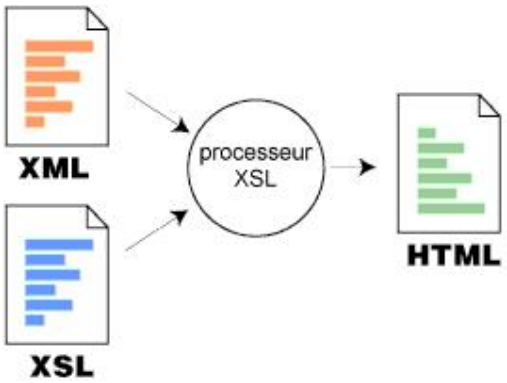
qui doit vous être fourni sous forme de fichier texte (ainsi que les fichiers images), réaliser le fichier XML accompagné de sa feuille de style afin dôobtenir le résultat de la page suivante :



Solution fournie sur le support SOLUTION.

XSL

Les spécifications du W3C traduites en Français !) sont sur http://xmlfr.org/w3c/TR/xslt/



« Le schéma présenté ci-dessus montre que la sortie produite est un document HTML. Mais ceci n'est pas restrictif. En effet, si aujourd'hui le seul processeur (ou "parser") XSL disponible (MSXSL) se limite à produire du HTML, on peut très bien penser que d'ici quelques temps des processeurs vont fleurir générant qui du RTF, qui du .doc, selon la finalité du document à produire... »
(http://www.gerbeaud.com/creation/xmlxsl/lesbases.htm)

EN TRAVAUX …  *EN TRAVAUX …*

CHAPITRE III : Les attributs

Il est possible d'ajouter des propriétés à un élément particulier en lui affectant un attribut, c'est-à-dire une paire clé/valeur.
Lôintérêt de ces informations supplémentaires est que celles-ci nôapparaissent pas à lôaffichage.
Elles peuvent servir en complément du contenu ou bien uniquement pour servir aux feuilles de style.
Ainsi avec XML la syntaxe pour définir un attribut est la suivante:
<! ATTLIST Elément Attribut Type >
Type représente le type de donnée de l'attribut, il en existe trois:

- Littéral: il permet d'affecter une chaîne de caractères à un attribut. Pour déclarer un tel type il faut utiliser le mot clé CDATA
- Enumération: cela permet de définir un liste de valeurs possibles pour un attribut donné, afin de limiter le choix de l'utilisateur. La syntaxe de ce type d'attribut est:

<! ATTLIST Elément Attribut (Valeur1 | Valeur2 | ...) >

Pour définir une valeur par défaut il suffit de faire suivre l'énumération par la valeur désirée entre guillemets:

<! ATTLIST Elément Attribut (Valeur1 | Valeur2) "valeur par défaut" >

- Atomique: il permet de définir un identifiant unique pour chaque élément

Enfin chacun de ces types d'attributs peut-être suivi d'un mot clé particulier permettant de spécifier le niveau de nécessité de l'attribut:

#IMPLIED signifie que l'attribut est optionnel, c'est-à-dire non obligatoire
#REQUIRED signifie que l'attribut est obligatoire
#FIXED signifie que l'attribut sera affecté d'une valeur par défaut s'il n'est pas défini. (Il doit donc être suivi dôune valeur entre guillemets)

Ainsi on pourra avoir une déclaration d'attribut du type:
<! ATTLIST PC IDpc ID #REQUIRED tailledisque(10|20|30)"10" >
Ce qui signifie que l'on affecte à l'élément PC deux attributs IDpc et tailledisque.
Le premier attribut est de type atomique, il s'agit d'un identifiant unique obligatoire.
L'élément tailledisque peut être soit 10,20,30.Sachant que 10 sera affecté par défaut.

EN TRAVAUX ...  *EN TRAVAUX ...*

CHAPITRE IV : Le D.O.M.

« Vous êtes un développeur en Visual Basic® et vous recevez des données sous la forme d'un document XML (eXtensible Markup Language). Vous souhaitez maintenant extraire les informations du document XML et les intégrer dans vos solutions Visual Basic. Vous pourriez bien sûr écrire vous-même le code destiné à analyser le contenu du fichier XML, qui est un simple fichier texte. Cependant, cette méthode n'est pas très productive et ne tient pas compte d'un des aspects essentiels du XML, à savoir qu'il permet de représenter des données de manière structurée.

L'utilisation d'un analyseur XML constitue une meilleure solution pour extraire les informations des fichiers XML. Un analyseur XML est tout simplement un logiciel qui lit un fichier XML et en récupère les données. En tant que développeur Visual Basic, vous souhaitez utiliser un analyseur qui gère le DOM (Document Object Model - Modèle objet de documents) XML. Le DOM définit un ensemble standard de commandes que les analyseurs doivent intégrer afin que vous puissiez accéder au contenu des documents HTML et XML depuis vos programmes. Un analyseur XML qui prend en charge le DOM extrait les données d'un document XML et les expose à l'aide d'un ensemble d'objets que vous exploitez.

Que signifie exactement DOM ?
Un DOM pour XML est un modèle objet qui expose le contenu d'un document XML. La spécification Document Object Model (DOM) Level 1 Specification du W3C définit actuellement ce qu'un DOM doit exposer comme propriétés, méthodes et événements. L'implémentation de DOM par Microsoft gère entièrement le standard défni par W3C et possède des fonctionnalités supplémentaires qui facilitent l'utilisation des fichiers XML à partir de vos programmes.

Pour utiliser le DOM XML, vous devez créer une instance d'un analyseur XML. Pour ce faire, Microsoft expose le DOM XML grâce à un ensemble d'interfaces COM standard figurant dans Msxml.dll, qui contient la bibliothèque de types et le code d'implémentation à utiliser avec les documents XML. Si vous employez un client script, par exemple VBScript dans Internet Explorer, vous activez le DOM à l'aide le la méthode CreateObject qui permet de créer une instance de l'objet Parser.

Set objParser = CreateObject("Microsoft.XMLDOM")
Si vous vous servez de VBScript à partir d'une page ASP (Active Server Page), vous devez utiliser Server.CreateObject.

Set objParser = Server.CreateObject("Microsoft.XMLDOM")
Si vous utilisez Visual Basic, vous pouvez accéder à DOM en établissant une référence à la bibliothèque de types MSXML, fournie dans Msxml.dll (DLL installée à partie de IE 5.0). Pour utiliser MSXML à partir de Visual Basic 6.0 :

Ouvrez la boîte de dialogue Projet - Références.
Sélectionnez Microsoft XML, version 2.0 dans la liste des objets COM disponibles. Si vous ne trouvez pas cet élément, vous devez obtenir la bibliothèque MSXML.
Vous pouvez alors créer une instance de l'objet Parser.
Dim xDoc As MSXML.DOMDocument Set xDoc = New MSXML.DOMDocument
Une fois la référence à la bibliothèque de types activée dans votre projet Visual Basic, lancez l'analyseur, chargez un document et exploitez-en les données.

À ce stade, vous pouvez vous demander : sur quoi suis-je en train de travailler ? Si vous ouvrez la bibliothèque MSXML et examinez son modèle objet à l'aide de l'Explorateur d'objets Visual Basic 6.0, vous constatez que le modèle objet est plutôt riche. Cet article explique comment accéder à un document XML à l'aide de la classe DOMDocument et de l'interface IXMLDOMNode

Pour charger un document XML, vous devez d'abord créer une instance de la classe DOMDocument :

```
Dim xDoc As MSXML.DOMDocument

Set xDoc = New MSXML.DOMDocument
```

Une fois que vous avez obtenu une référence valide, ouvrez un fichier à l'aide de la méthode Load. L'analyseur MSXML peut charger les documents XML à partir du disque local, via le réseau à l'aide de références UNC ou via une URL.

Pour charger un document à partir d'un disque, utilisez le type de structure suivant en employant la méthode Load :

```
If xDoc.Load("C:\My Documents\cds.xml") Then

' Le document a été chargé avec succès.

' Maintenant faites quelque chose d'intéressant.

Else

' Impossible de charger le document

End If
```

Vous devez libérer la référence de l'objet vers le document quand vous en avez terminé avec celui-ci. L'analyseur MSXML n'offre pas de méthode Close explicite. La meilleure solution consiste à définir de manière explicite la référence sur Nothing.

```
Set xDoc = Nothing
```

Par défaut, l'analyseur charge un fichier de manière asynchrone. Vous pouvez modifier ce comportement à l'aide de la propriété booléenne Async du document. Vous devez examiner la propriété ReadyState d'un document pour vous assurer qu'un document est prêt avant l'examen de son contenu. La propriété ReadyState peut renvoyer 1 valeur parmi les 5 énumérées ci-dessous :

Unitialized	: le chargement n'a pas commencé.	Valeur 0
Loading	: la méthode Load est en cours d'exécution.	Valeur 1
Loaded	: la méthode Load est achevée.	Valeur 2
Interactive	: une proportion suffisante du DOM est disponible pour l'examen en lecture seule et les données n'ont été que partiellement analysées.	Valeur 3
Completed :	les données sont chargées, analysées et disponibles pour les opérations de lecture/écriture.	Valeur 4

L'analyseur MSXML expose les événements que vous pouvez utiliser lors du chargement de documents volumineux afin de suivre l'état du processus de chargement. Ces événements sont également utiles lors du chargement asynchrone d'un document à partir d'une URL via Internet.

Pour ouvrir un fichier à partir d'une URL, vous devez spécifier l'emplacement du fichier à l'aide d'une URL complète. Vous devez notamment ajouter le préfixe http://.

Voici un exemple de chargement d'un fichier à partir d'une URL :

```
xDoc.async = False

If xDoc.Load("http://www.develop.com/hp/brianr/cds.xml") Then

' Le document a été chargé avec succès.

' Maintenant faites quelque chose d'intéressant.

Else

' Impossible de charger le document

End If
```

En définissant la propriété Async du document sur False, l'analyseur ne rend pas la main à votre code tant que le document n'est pas entièrement chargé et son exploitation possible. Si vous laissez la propriété définie sur True, vous devez soit examiner la propriété ReadyState avant d'accéder au document soit utiliser les événements de DOMDocument pour que le code reçoive une notification lorsque le document est prêt.

Le chargement d'un document peut échouer pour plusieurs raisons. Il arrive souvent que le nom du document transmis à la méthode Load soit incorrect. Il arrive aussi que le document XML soit incorrect.

Par défaut, l'analyseur MSXML validera votre document à l'aide d'une DTD ou d'un schéma si l'un des deux est spécifié dans le document. Vous pouvez indiquer à l'analyseur de ne pas valider le document en définissant la propriété ValidateOnParse de la référence d'objet DOMDocument avant d'invoquer la méthode Load.

```
Dim xDoc As MSXML.DOMDocument

Set xDoc = New MSXML.DOMDocument
```

```
xDoc.validateOnParse = False

If xDoc.Load("C:\My Documents\cds.xml") Then

' Le document a été chargé avec succès.

' Maintenant faites quelque chose d'intéressant.

Else

' Impossible de charger le document

End If
```

Vous devez savoir que la désactivation de la validation de l'analyseur n'est pas une bonne solution pour les applications de production. Un document incorrect peut provoquer l'échec du programme pour plusieurs raisons. Dans le meilleur des cas, il fournira des données incorrectes aux utilisateurs.

Vous pouvez demander à l'analyseur des informations sur l'échec, indépendamment du type d'erreur, en accédant à l'objet ParseError. Définissez une référence à l'interface IXMLDOMParseError du document afin d'utiliser les propriétés de l'objet ParseError. L'interface IXMLDOMParseError expose sept propriétés que vous pouvez utiliser pour rechercher la cause de l'erreur.

L'exemple ci-après affiche une boîte de message et toutes les informations d'erreur disponibles à partir de l'objet ParseError.

```
Dim xDoc As MSXML.DOMDocument
Set xDoc = New MSXML.DOMDocument

If xDoc.Load("C:\My Documents\cds.xml") Then
' Le document a été chargé avec succès.
' Maintenant faire quelque chose d'intéressant.
Else
' Impossible de charger le document
Dim strErrText As String
Dim xPE As MSXML.IXMLDOMParseError
' Obtient l'objet ParseError
Set xPE = xDoc.parseError

With xPE
strErrText = "Your XML Document failed to load " & _
"due the following error." & vbCrLf & _
"Error #: " & .errorCode & ": " & xPE.reason & _
"Line #: " & .Line & vbCrLf & _
"Line Position: " & .linepos & vbCrLf & _
"Position In File: " & .filepos & vbCrLf & _
"Source Text: " & .srcText & vbCrLf & _
"Document URL: " & .url
End With

MsgBox strErrText, vbExclamation
End If

Set xPE = Nothing
```

Vous pouvez utiliser les informations exposées par l'objet ParseError afin d'afficher ces informations pour l'utilisateur, les consigner dans un fichier d'erreur ou essayer de corriger l'erreur.

Extraction des informations d'un document XML
Une fois le document chargé, l'étape suivante consiste à en extraire les informations. Même si l'objet de document est important, vous utiliserez principalement l'interface IXMLDOMNode. L'interface IXMLDOMNode vous servira à lire et à écrire dans chaque élément de noeuds. Avant de poursuivre, vous devez comprendre qu'il existe 13 types de noeuds pris en charge par l'analyseur MSXML. Le tableau suivant répertorie les types de noeuds les plus fréquents.

Type de noeud DOM	***Exemple***
NODE_ELEMENT	*<artist type="band">Offspring</artist>*
NODE_ATTRIBUTE	*<artist type="band">Offspring</artist>*
NODE_TEXT	*<artist type="band">Offspring</artist>*
NODE_PROCESSING_INSTRUCTION	*<?xml version="1.0"?>*
NODE_DOCUMENT_TYPE	*<!DOCTYPE compactdiscs SYSTEM "cds.dtd">*

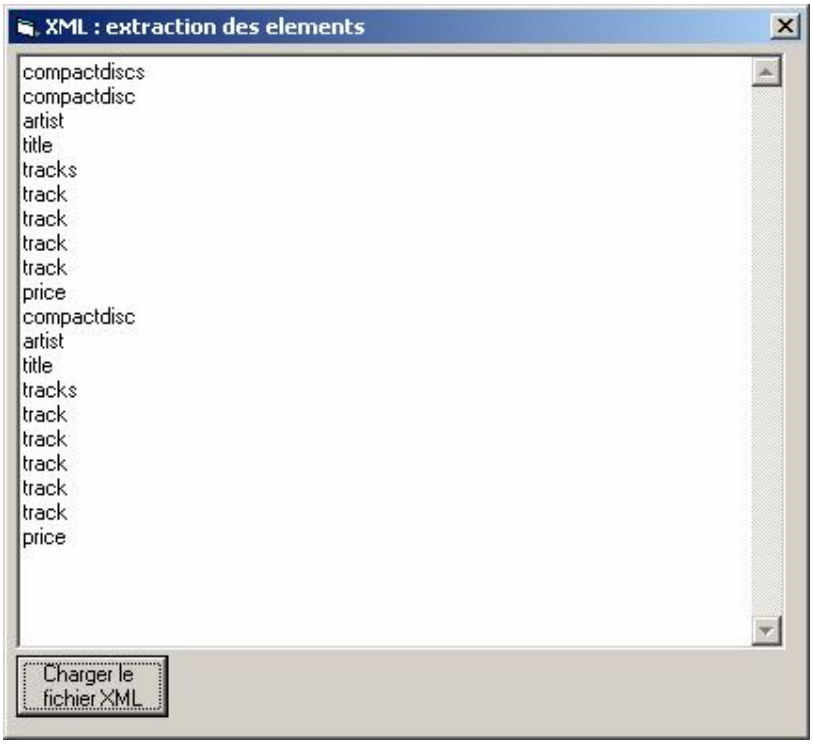
Vous pouvez accéder au type de noeud au moyen de deux propriétés exposées par l'interface IXMLDOMNode. La propriété NodeType expose une énumération des éléments DOMNodeType (dont certains sont répertoriés dans le tableau précédent). En outre, vous pouvez utiliser NodeTypeString afin de récupérer une chaîne de texte pour le type de noeud.

Une fois que vous avez une référence à un document, vous pouvez commencer à parcourir la hiérarchie nodale. À partir de la référence de document, vous pouvez accéder à la propriété ChildNodes, qui donne un point d'entrée vertical vers tous les noeuds du document. La propriété ChildNodes expose IXMLDOMNodeList, qui gère la structure For/Each de Visual Basic. Vous pouvez ainsi énumérer chacun des noeuds de la propriété ChildNodes. De plus, la propriété ChildNodes expose une propriété Level qui renvoie le nombre de noeuds enfants existants.

L'objet de document n'est pas le seul à exposer une propriété ChildNodes : chaque noeud l'expose également, ce qui permet, associé à la propriété HasChildNodes de IXMLDOMNode, de parcourir plus facilement la hiérarchie nodale en examinant les éléments, les attributs et les valeurs.

Il faut garder à l'esprit la relation parent-enfant entre l'élément d'un document et la valeur de cet élément. Par exemple, dans le document XML des disques compacts, l'élément <title> expose un titre de chanson. Pour extraire la valeur de l'élément <title> vous devez rechercher les noeuds de type NODE_TEXT. Une fois que vous avez trouvé un noeud avec des données intéressantes, vous pouvez examiner les attributs et même accéder à son noeud parent grâce à la propriété ParentNode. » *(extrait d'un article de MSDN on line de Brian Randell ; DevelopMentor)*

Voici un programme exploitant le DOM XML pour en extraire les élements :

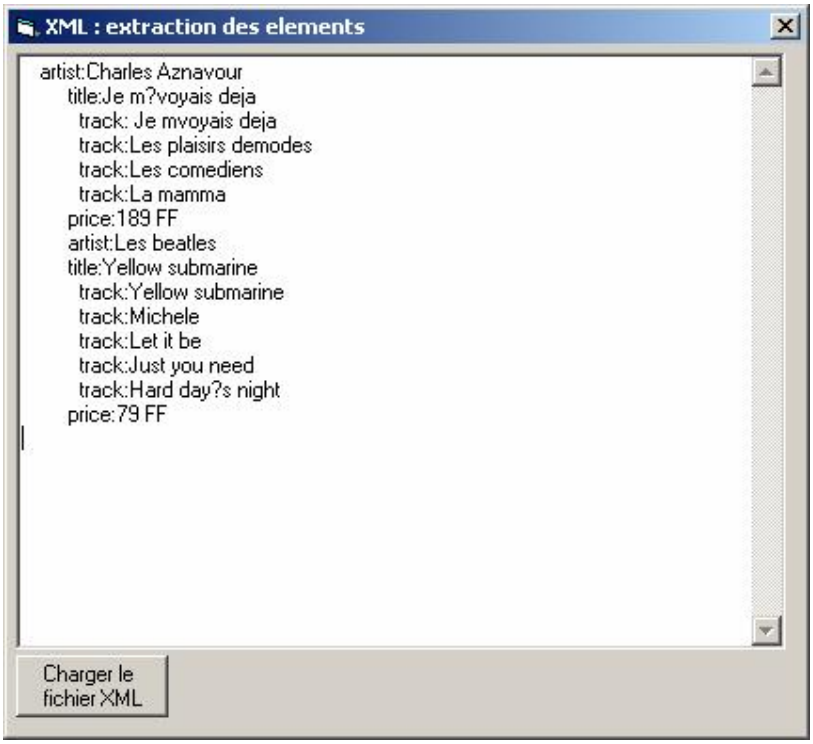



```
Option Explicit

Dim oDocXML As New MSXML.DOMDocument
'Reference : Microsoft XML, version 2.0
'         ( c:\winNt\system32\msxml.dll )
'Dim WithEvents oDocXML As MSXML.DOMDocument  'Si on veut trapper les evenements


Private Sub cmdChargerXML_Click()
CommonDialog1.Filter = "XML (*.xml)|*.xml|All (*.*)|*.*"
CommonDialog1.ShowOpen

On Error Resume Next  'pas beau mais indispensable

'Si l'instanciation via "new" ( declaration ) était omise, alors il faudrait:
'Set oDocXML = CreateObject("Microsoft.XMLDOM")
oDocXML.Load CommonDialog1.FileName 'Chargement du fichier

Dim oUnNoeud  As MSXML.IXMLDOMNode  'Un variant pourrait faire l'affaire
While oDocXML.readyState <> 4       ' READYSTATE_COMPLETE
   DoEvents                         ' mesure de précaution
Wend

Dim i As Integer
For i = 0 To oDocXML.childNodes.length
   If oDocXML.childNodes(i).nodeType = NODE_ELEMENT Then
      ParcoursElement oDocXML.childNodes(i)
   End If
Next i
End Sub

Private Sub ParcoursElement(ByVal p_oUnNoeud As MSXML.IXMLDOMNode)

Text1.Text = Text1.Text & p_oUnNoeud.nodeName & vbCrLf

If p_oUnNoeud.hasChildNodes Then  'Si le noeud a des enfants
   Dim i As Integer
   For i = 0 To p_oUnNoeud.childNodes.length - 1
         If p_oUnNoeud.childNodes(i).nodeType = NODE_ELEMENT Then
            ParcoursElement p_oUnNoeud.childNodes(i)
```

```
        End If
    Next i
End If
End Sub
```

Exercice

Adapter ce source pour obtenir le résultat suivant (c.a.d les élements avec du contenu)



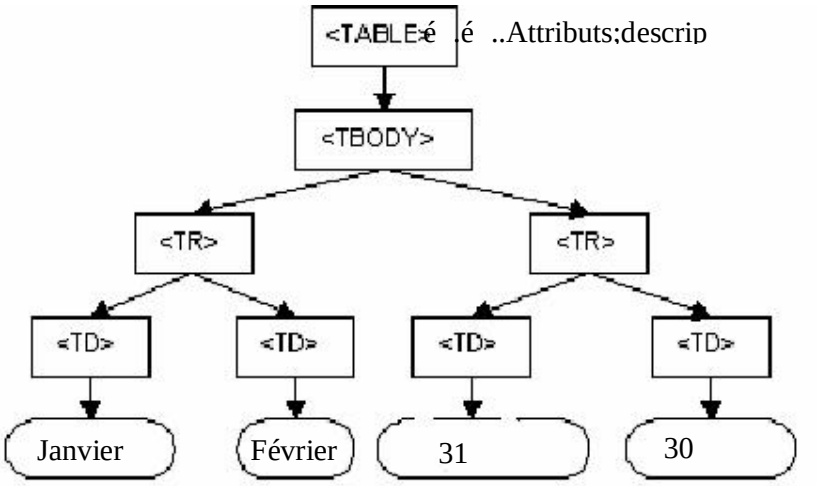
Manipulation du D.O.M Via scripting

Rappel :
Un DOM est un objet qui range soigneusement le contenu de pages dans une structure sous forme arborescente.

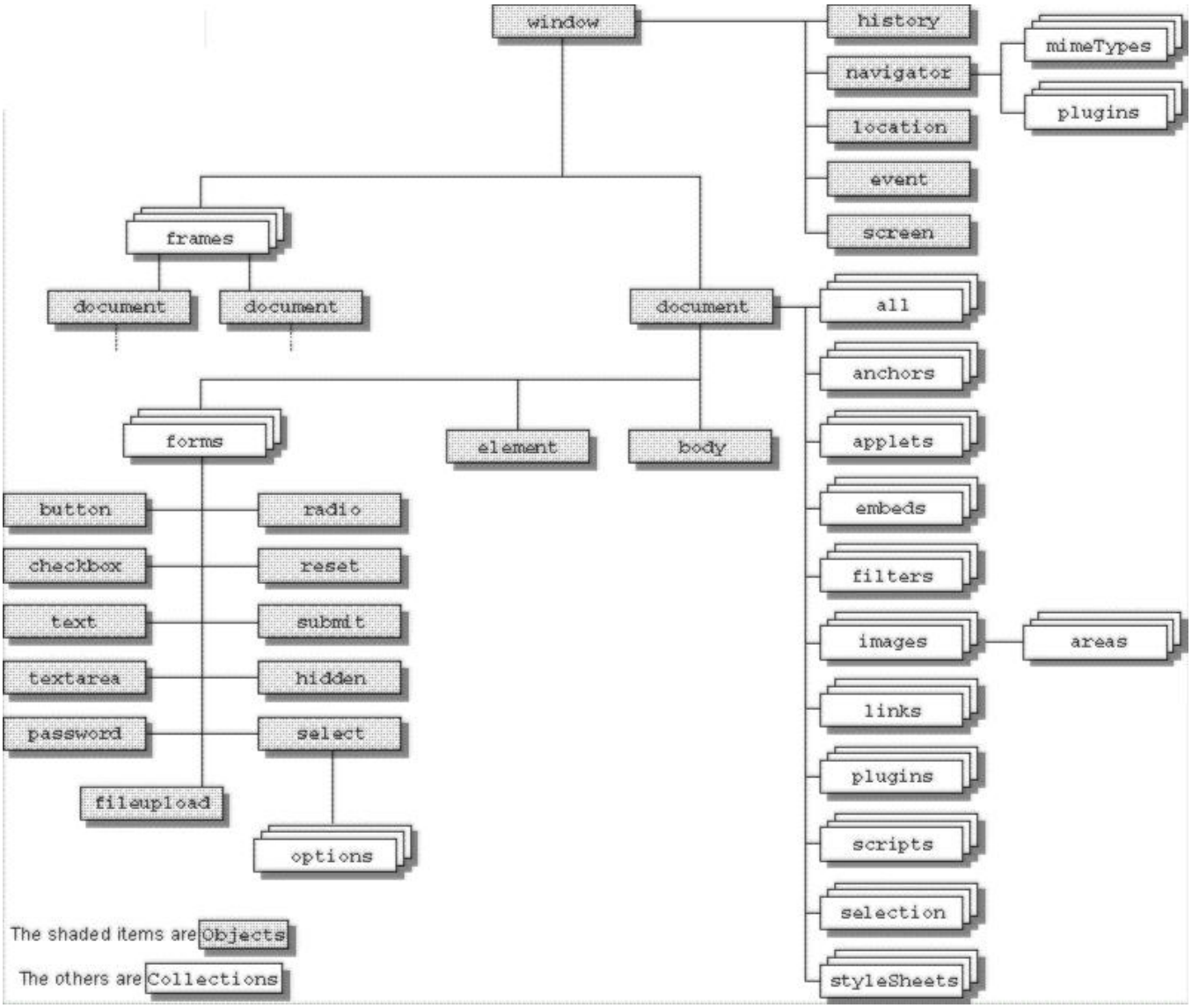
Par exemple si nous avons le source HTML suivant :

```
<TABLE border="1">
    <TBODY>
        <TR>
            <TD>Janvier</TD>
            <TD>Février</TD>
        </TR>
        <TR>
            <TD>31</TD>
            <TD>30</TD>
        </TR>
    </TBODY>
</TABLE>
```

Sa représentation via le DOM peut s'imaginer comme ça :



DESCRIPTION COMPLETE DU DOM HTML

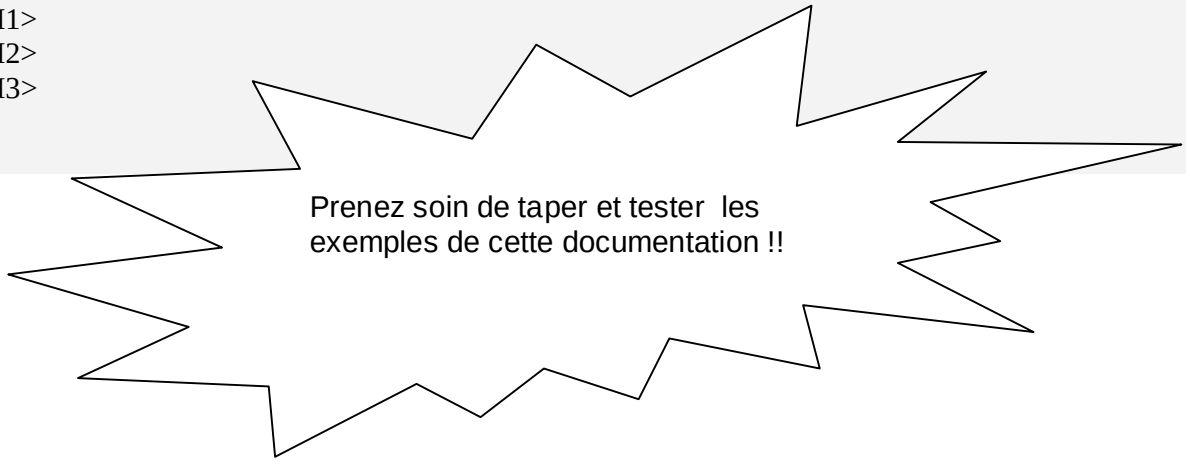


Voici un exemple de script qui utilise le DOM HTML :

```
<html>
<head>

<SCRIPT LANGUAGE="JavaScript">

 function OuEstLeClick()
   {
     window.alert("Le click est sur " + window.event.srcElement.tagName);
   }

</SCRIPT>

</HEAD>

<BODY onclick="OuEstLeClick()">

<H1>H1</H1>
<H2>H2</H2>
<H3>H3</H3>
</body>
</html>
```

Et le même avec VBScript :

```
<html>
<head>

<SCRIPT LANGUAGE="VbScript">

 sub OuEstLeClick()
    window.alert("Le click est sur " + window.event.srcElement.tagName)
 end sub
 </SCRIPT>

</HEAD>

<BODY onclick="OuEstLeClick()">

<H1>H1</H1>
<H2>H2</H2>
<H3>H3</H3>
</body>
</html>
```



Lorsque le navigateur reçoit un flot de données, il le range soigneusement dans des collections rattachées à on objet ; le Document Object Model.
Le Document Object Model est la tentative de représenter toutes les parties significatives d'une page WWW affichée comme une hiérarchie d'objets.

Il est particulièrement intéressant de mettre une source de données XML dans du HTML.
On pourra ainsi en exploiter facilement le contenu ; soit pour jouer sur la présentation (en choisissant d'afficher tout ou partie des données) soit pour faire varier le contenu.

Comment insérer du source XML dans du HTML ?
Essayons :

```
<html>
<body>
Ceci est du texte
<infos_PC>
        <description>
           <nom>PC Home</nom>
           <prix_unitaire>5999</prix_unitaire>
           <fournisseur>Sulyo</fournisseur>
           <particularites>Base + lecteur DVD</particularites>
        </description>
        <description>
           <nom>PC Multimedia</nom>
           <prix_unitaire>7999</prix_unitaire>
           <fournisseur>Fukoji</fournisseur>
           <particularites>PC HOME + ecran 19 + carte son</particularites>
        </description>
</infos_PC>
<body>
</html>
```

Le résultat est que le browser affiche le contenu des balises XML.
Si nous voulons pouvoir manipuler cette source de données de façon plus fine, il est nécessaire d'indiquer au navigateur qu'il s'agit bien d'un **îlot de données** XML ; ce qui donne

```
<html>
<body>
 Ceci est du texte
<XML>
<infos_PC>
        <description>
           <nom>PC Home</nom>
           <prix_unitaire>5999</prix_unitaire>
           <fournisseur>Sulyo</fournisseur>
           <particularites>Base + lecteur DVD</particularites>
        </description>
        <description>
           <nom>PC Multimedia</nom>
           <prix_unitaire>7999</prix_unitaire>
           <fournisseur>Fukoji</fournisseur>
           <particularites>PC HOME + ecran 19 + carte son</particularites>
        </description>
</infos_PC>
</XML>
<body>
</html>
```

ÎLOT DE DONNEES

(source html_xml1.htm)

Une façon plus élégante et plus souple consiste à indiquer un lien sur le fichier xml plutôt que de l'insérer « en dur » dans la page (ce principe est connu depuis que nous avons étudié les DTD et les CSS).
La syntaxe est <XML **SRC**="NomFichierXML.xml">

Cette seconde forme d'îlot de données se conforme davantage à la philosophie XML consistant à maintenir les données séparées des informations de formatage et de traitement.
Ce qui donne pour l'exemple précédant :

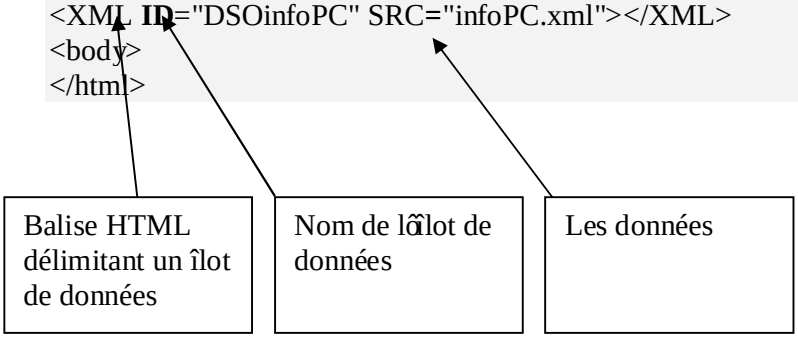
```
<html>
<body>
Ceci est du texte
<XML SRC="infoPC.xml"></XML>
<body>
</html>
```

(source html_xml2.htm)

Le navigateur va ranger le contenu du fichier XML dans un objet appelé DSO pour Data Source Object que l'on traduira par "îlot de données".

Comme il faudra pouvoir accéder à cet îlot il est nécessaire de lui donner un identificateur ; ce qui donnera :

```
<html>
<body>
Ceci est du texte
<XML ID="DSOinfoPC" SRC="infoPC.xml"></XML>
<body>
</html>
```

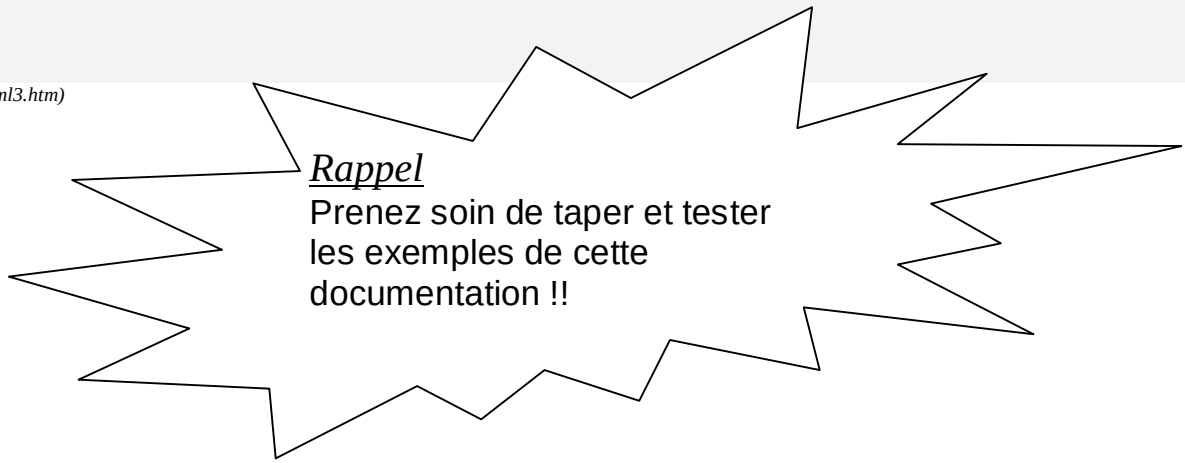


Digression :
Il est possible de lier d'autres sources de données à du HTML.
Pour des fichiers textes on utilisera le contrôle DTC ("clsid:333C7BC4-460F-11D0-BC04-0080C7055A83") livré depuis IE4.
Pour des bases de données relationnelles, utiliser RDS ou JDBC Java Source Applet.
(Plus d'infos sur http://icapc35.epfl.ch/2001/howtos/selfhtml/tfbb.htm)

Puis pour manipuler les données XML il suffira de parcourir le DSO que lôon peut considérer comme une sorte de recordset.

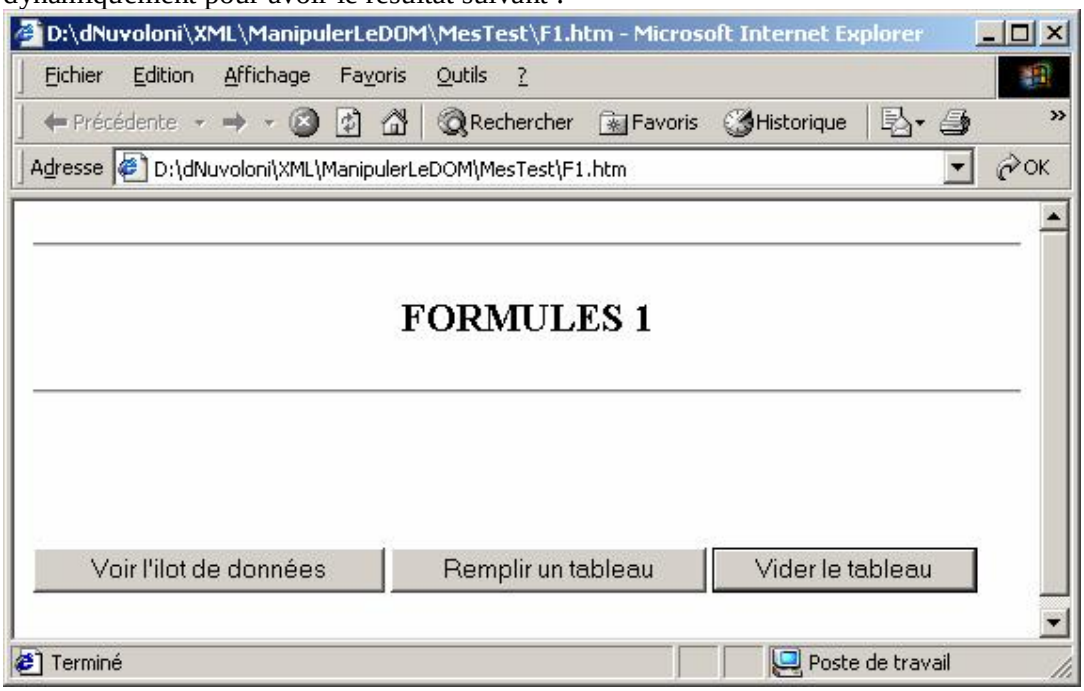
```
<html>
	<SCRIPT LANGUAGE="VbScript">

	  sub MontrerXML()
		xmlDoc.recordset.movefirst

		while not(xmldoc.recordset.eof)
		  dsoXML=dsoXML + "Nom->" + xmlDoc.recordset("Nom").value + "<br>"
		  dsoXML=dsoXML + "prix_unitaire->" + xmlDoc.recordset("prix_unitaire").value + "<br>"
		  dsoXML=dsoXML + "fournisseur->" + xmlDoc.recordset("fournisseur").value + "<br>"
		  dsoXML=dsoXML + "<br><br>"
		  xmlDoc.recordset.movenext
		wend
		'On accroche la branche du D.O.M "CibleDansHTML"
		oUneTable=document.getElementsByName("CibleDansHTML")
		oUneTable.innerHTML=  dsoXML

		' !!! Autre syntaxe  !!!
		'document.all.item("Tableau").innerHTML=  "<Table
border='1'><tr><td>xxx</td><td>yyy</td></tr></Table>"
	  end sub

	</SCRIPT>

<body>
Voir le contenu du source XML
<XML ID="xmlDoc" SRC="infoPC.xml"></XML>
	<DIV ID="CibleDansHTML">
	</DIV>
	<br>
	<br>
	<input type="button" value="Voir l'ilot de données" onclick="MontrerXML()">

<body>

</html>
```

(source html_xml3.htm)

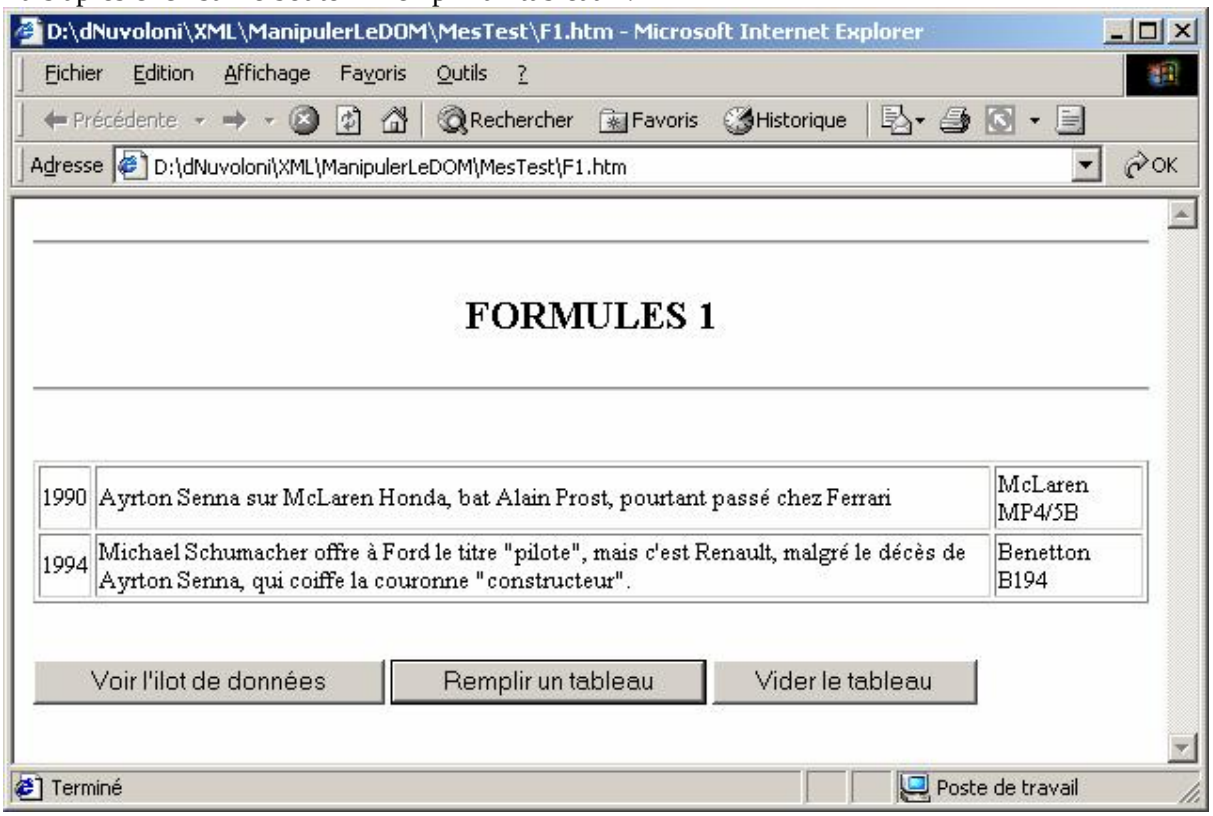


TP DSO_1 :

Réaliser le source HTML contenant un îlot de données XML (F1.xml) afin de remplir un tableau dynamiquement pour avoir le résultat suivant :



Puis après click sur le bouton "Remplir un tableau" :



Solution hors support (solution.doc).

Une autre façon dôexploiter un source XML consiste à jouer avec le DATABINDING, côest à dire de relier des balises à des données sans programmation.

```
<html>

<body>
<b>"Binding" XML / HTML</b>
<br><br><br>
<XML ID="xmlDoc" SRC="infoPC.xml"></XML>
<TABLE Border="1">
<THEAD>
  <TH><font color="#008000">nom</font> </TH>
  <TH><font color="#008000">prix_unitaire</font></TH
  <TH><font color="#008000">fournisseur</font></TH>
  <TH><font color="#008000">particularites</font></TH
<TR>
<TD>  <span DataSrc="#xmlDoc" DataFld="nom" </span></TD>
<TD>  <span DATASRC="#xmlDoc" DataFld="prix_unitaire" </span></TD>
<TD>  <span DataSrc="#xmlDoc" DataFld="fournisseur" </span></TD>
<TD>  <span DATASRC="#xmlDoc" DataFld="particularites" </span></TD>

</TR>
</TABLE>
        <br>
        <br>
<BUTTON onclick="xmlDoc.recordset.movefirst()" id=button1 name=button1><<</BUTTON>
<BUTTON onclick="xmlDoc.recordset.moveprevious()"  id=button2 name=button2><</BUTTON>
<BUTTON onclick="xmlDoc.recordset.movenext()" id=button3 name=button3>></BUTTON>
<BUTTON onclick="xmlDoc.recordset.movelast()" id=button4 name=button4>>></BUTTON>

<!-- Autre syntaxe pour le bouton -->
<!-- <input type="button" value=">>" onclick="xmlDoc.recordset.movelast()"> -->

</html>
```

(source html_xml4.htm)

Les DataFld sont considérés comme des colonnes dôune base de données

Une possibilité encore plus simple revient à relier la table directement sur le DSO et à laisser faire IE :

```
<html>

<body>
<b>"Binding" XML / HTML</b>
<br><br><br>
<XML ID="xmlDoc" SRC="infoPC.xml"></XML>
<TABLE DataSrc="#xmlDoc" Border="1">
<THEAD>
  <TH><font color="#008000">nom</font> </TH>
  <TH><font color="#008000">prix_unitaire</font></TH>
  <TH><font color="#008000">fournisseur</font></TH>
  <TH><font color="#008000">particularites</font></TH>
 </THEAD>
<TR>
<TD>  <span DataFld="nom" </span></TD>
<TD>  <span DataFld="prix_unitaire" </span></TD>
<TD>  <span DataFld="fournisseur" </span></TD>
<TD>  <span DataFld="particularites" </span></TD>

</TR>
</TABLE>
        <br>
        <br>

</html>
```

(source html_xml5.htm)

Même si lôélément table ne définit quôune seule ligne, lorsque le navigateur affiche la table, il répète lôélément TR pour chaque enregistrement du document XML.

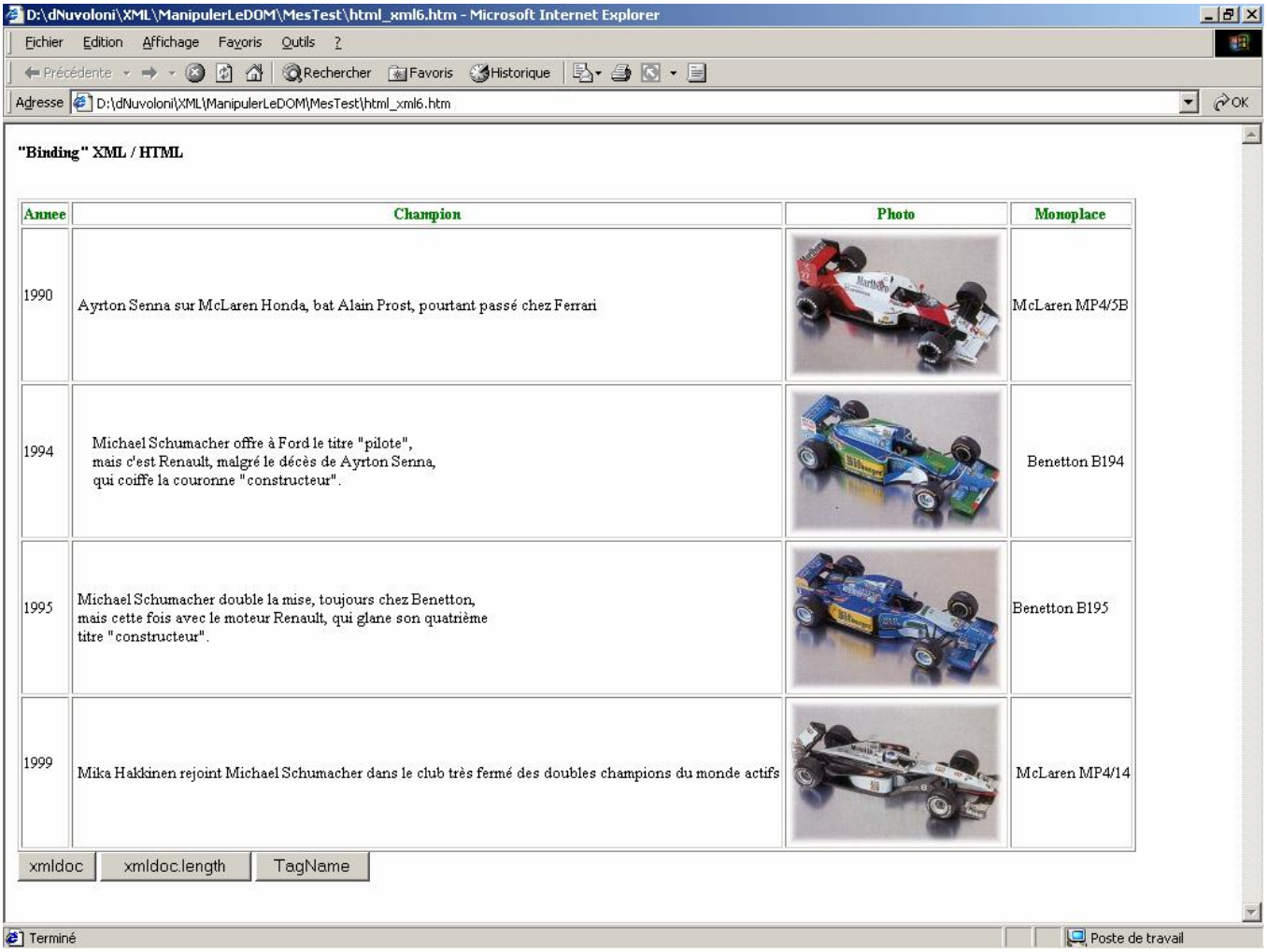
TP DSO_2 :

Comment modifier le source xml (f1.xml) pour que le fichier suivant :

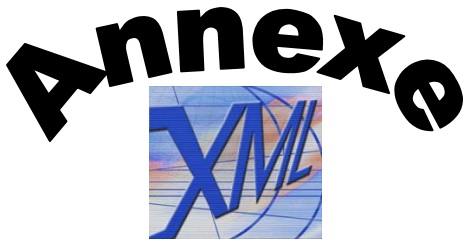
```
<html>

<body>
<b>"Binding" XML / HTML</b>
<br><br><br>
<XML ID="xmlDoc" SRC="F1.xml"></XML>
<TABLE DataSrc="#xmlDoc" Border="1">
<THEAD>
  <TH><font color="#008000">Annee</font> </TH>
  <TH><font color="#008000">Champion</font></TH>
  <TH><font color="#008000">Photo</font></TH>
  <TH><font color="#008000">Monoplace</font></TH>
 </THEAD>
<TR>
<TD>    <span DataFld="Annee" </span></TD>
<TD>    <span DataFld="Champion" </span></TD>
<TD >   <IMG  DataFld="Photo"></TD>
<TD>    <span DataFld="Monoplace" </span></TD>

</TR>
</TABLE>

<input type="button" value="xmldoc"
onclick=alert(xmlDoc.documentElement.childNodes.item(1).text) id=button1 name=button1>
<input type="button" value="xmldoc.length"
onclick=alert(xmlDoc.documentElement.childNodes.length) id=button2 name=button2>
<input type="button" value="TagName"
onclick=alert(xmlDoc.documentElement.getElementsByTagName("Photo").item(1).text) id=button2 name=button2>

            <br>
            <br>
</html>
```

(html_xml6.htm)

Affiche lôécran suivant ?



Solution dans « solutions.doc »



Définitions

- **DTD** : **D**ocument **T**ype **D**efinition
- **HTML** : **H**yperText **M**arkup **L**anguage
- **SGML** : **S**tandard **G**eneralized **M**arkup Language
- **XML** : e**X**tensible **M**arkup **L**anguage
- **XLL** : e**X**tensible **L**ink **L**anguage
- **XQL** : e**X**tensible **Q**uery **L**anguage
- **XSL** : e**X**tensible **S**tyle **L**anguage
- **CSS** : **C**ascading **S**tyle **S**heets
- **SOAP** : **S**imple **O**bject **A**ccess **P**rotocol

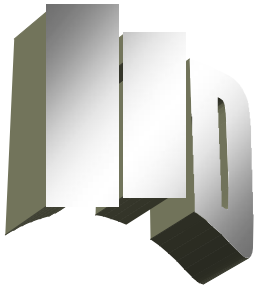
Glossaire XML sur

☞ http://www.xmltechno.com/glossaire/index.cfm

Contenu d'une DTD

déclaration d'élément : ELEMENT
<!ELEMENT element1 (element2, element3)>
element1 est composé de element2 suivi de element3

<!ELEMENT element1 (element2 | element3)>
element1 est composé de element2 ou de element3



<!ELEMENT element1 (element2)?>
element1 est composé de 0 ou 1 element2

<!ELEMENT element1 (element2)*>
element1 est composé de 0 ou plusieurs element2

<!ELEMENT element1 (element2)+>
element1 est composé de 1 ou plusieurs element2

<!ELEMENT element1 (#PCDATA)>
element1 est composé de caractères "parsable", donc s'il y a des entités, elles sont résolues.

<!ELEMENT element1 EMPTY>
element1 est toujours vide de contenu

<!ELEMENT element1 ANY>
element1 contient n'importe quels éléments définis dans la DTD ou (#PCDATA)
déclaration d'entités : ENTITY et notations : NOTATION

<!ENTITY nom-entité "chaine-de-remplacement"> entité interne
<!ENTITY nom-entité SYSTEM "URI-de-remplacement"> entité externe

déclaration d'attribut :
= triplet nom_d'attribut type valeur_par_défaut
Un attribut est plus qu'un élément "texte" : le type est plus précis, et sa valeur par défaut peut être précisée. Voici quelques possibilités :
syntaxe :
<!ATTLIST nom-de-l'élément nom_d'attribut type valeur_par_défaut>
<!ATTLIST nom-de-l'élément nom_d'attribut type valeur_par_défaut>
nom_d'attribut type valeur_par_défaut
.... >
Type d'attributs
(valeur1 | valeur2 | ...) liste de valeurs possibles pour l'attribut
CDATA texte non "parsé"
ID un identifiant unique
IDREF une référence à un identifiant unique du document
ENTITY un nom d'entité de la DTD.
NOTATION le nom d'une notation (référençant une entité non XML)
...
Valeur par défaut de l'attribut
#REQUIRED la valeur de l'attribut doit obligatoirement être indiquée
#IMPLIED la valeur peut être omise, mais les applications traitant le document lui attribueront une valeur par défaut.
"defaultValue" valeur par défaut, si elle est omise
#FIXED "fixedValue" constante.

F.A.Q

(Extrait de http://www.gutenberg.eu.org/pub/GUTenberg/publications/HTML/FAQXML/faqxml-fr.html#francais)

Qu'est-ce-que XML ?
XML signifie Extensible Markup Language (langage extensible de balisage.). Ce langage est dit extensible car, contrairement à HTML, il ne s'agit pas d'un format fixe. Son but est de permettre d'utiliser SGML sur le World Wide Web.
XML n'est pas un simple langage de balisage prédéfini. C'est un méta-langage -- un langage qui permet de définir d'autres langages -- qui vous permet de concevoir votre propre balisage. Un langage de balisage prédéfini comme HTML définit une façon de décrire les informations pour une certaine classe de documents : XML vous permet de définir vos propres langages de balisage personnalisés pour un ensemble de classes de documents. Ceci est possible parce que XML est écrit en SGML, le méta-langage international normalisé de balisage.

Qui est responsable de XML ?
XML est un projet duWorld Wide Web Consortium (W3C) et le développement de sa norme est supervisé par son Groupe de travail XML. Un Groupe d'intérêt spécial composé de collaborateurs admis par cooptation et d'experts issus de divers domaines a apporté commentaires et révisions via le courrier électronique.
XML est un format public : aucune société n'est propriétaire de son développement. La spécification v1.0 a été entérinée par le W3C (recommandation du 10 février 1998).

En quoi XML simplifie-t-il SGML tout en permettant de définir des types de documents personnalisés ?
Pour simplifier SGML, XML redéfinit certaines fonctions et certains paramètresSGML et supprime un grand nombre des fonctions les plus complexes et parfois les moins utilisées qui rendent l'écriture de programmes de traitement difficile (voir http://www.w3.org/TR/NOTE-sgml-xml-971215).
Même si XML garde toutes les fonctions de structure de SGML qui permettent de définir son propre type de document, XML introduit aussi une nouvelle classe de documents qui n'oblige pas à utiliser un type de document prédéfini (en gros, on peut définir son propre balisage à condition de suivre rigoureusement les règles syntaxiques).

Pourquoi ne continue-t-on pas à développer HTML ?
HTML est déjà surchargé de dizaines d'inventions de divers constructeurs intéressantes mais souvent incompatibles car il ne prévoit qu'une seule manière de décrire vos informations.

XML permettra à des groupes de personnes ou à des organisations de créer leurs propres langages de balisage personnalisés afin d'échanger des informations dans leurs domaines (musique, chimie, électronique, randonnée, finance, surf, géologie pétrolière, linguistique, cuisine, tricot, cartographie stellaire, histoire, ingénierie, élevage de lapins, mathématiques, etc).

HTML a atteint les limites de son utilité en tant que descripteur d'informations. Et même s'iil continue à jouer un rôle important pour le contenu qu'il représente aujourd'hui, de nombreuses autres applications nécessitent une infrastructure plus robuste et plus flexible.

Je suis habitué à utiliser HTML. Est-ce que je peux apprendre XML facilement ?

Oui, très facilement. Mais aujourd'hui, il manque toujours des cours, des outils simples et plus d'exemples de documents XML. Des documents XML bien formés peuvent paraître similaires à HTML excepté pour de petits mais très importants points de syntaxe.

La principale différence est que XML doit coller aux règles. Les navigateurs HTML en revanche vous permettent de créer du HTML dégradé car ils ignorent les informations erronées : avec XML, ou bien votre fichier est correct, ou bien ile ne marche pas !

Comment contrôler la présentation ?

L'utilisation d'une feuille de style est obligatoire en XML. Certains navigateurs offrent peut-être de simples styles par défaut pour des éléments courants comme <PARA> ou <LIST> contenant des <ITEM> ; mais en général une feuille de style permet à l'auteur de mieux contrôler la mise en page. Or, comme dans tout système où les fichiers peuvent être visualisés au hasard par des utilisateurs arbitraires, l'auteur ne peut pas connaître les ressources (polices par exemple) disponibles sur le système de l'utilisateur ; des précautions sont donc de rigueur.

La norme internationale de feuilles de style pour les documents SGML est DSSSL, Document Style and Semantics Specification Language (ISO 10179). Elle offre des langages du genre de Scheme pour les feuilles de style et la conversion de documents et est implémenté dans le logiciel de formatage Jade.
Les CSS (Cascading Stylesheet Specifications) offrent une syntaxe simple pour attribuer des styles aux éléments et ont été implémentées en partie dans certains navigateurs HTML.
Une nouvelle ébauche de (XSL) Extensible Style Language a été conçu pour une utilisation spécifique avec XML . Elle utilise la syntaxe XML (une feuille de style XSL est en fait un fichier XML) mais combine les fonctions de formatage de DSSSL et de CSS (HTML) et bénéficie déjà du soutien de nombreux distributeurs importants.

Sur le site Microsoft
http://www.microsoft.com/FRANCE/MSDN/Technologies/XML/info/info.asp?mar=/FRANCE/MSDN/Technologies/XML/info/QRXML.html

Dans la réalité, comment XML est-il utilisé ?

XML est utilisé dans un nombre incroyable d'applications, depuis la création et la documentation d'un site Web jusqu'à l'intégration d'une base de donnée en passant par la programmation distribuée. Voici quelques domaines dans lesquels XML a prouvé son utilité :

Les transmissions interentreprises.
Les données d'entreprise (factures, commandes, informations comptables et fiscales, etc.) sont transférées électroniquement entre les vendeurs au format XML. XML offre un certain nombre d'avantages par rapport à l'ancien format d'échange de données informatisées (EDI), en particulier la facilité d'écriture des transformations permettant de convertir le format des factures.

La programmation distribuée.
XML est parfaitement adapté à la création d'applications complexes à plates-formes multiples et permet d'utiliser vos serveurs Windows avec d'autres systèmes d'exploitation.

L'architecture des sites Web.
XML étant par nature un langage hiérarchique et distribué, les développeurs de sites Web l'utilisent de plus en plus dans l'architecture globale et dans les structures de navigation de leurs sites. Il est de plus en plus courant d'utiliser XML et XSLT pour écrire les tables de matières et les index, les structures XML permettant de suivre les informations utilisateur et l'état du site Web, les composants HTML et les conduits destinés à orienter les flux de données.

Communication avec les bases de données.
XML est de plus en plus apprécié comme moyen privilégié de communication avec les bases de données (qu'il s'agisse de récupérer des ensembles de données XML à partir d'une requête SQL ou de mettre à jour une base de données en utilisant un enregistrement XML). Là encore, nous bénéficions de la séparation des implémentations. En exprimant les données en XML, vous n'avez pas besoin de connaître la structure réelle de la base de données.

Gestion des documents.
La plupart des entreprises sont actuellement submergées par la paperasserie et cela ne pourra qu'empirer. XML est de plus en plus utilisé pour coder les documents en XML de façon à faciliter leur recherche ou à fournir un certain contexte d'annotation des documents, qui peuvent ainsi être référencés de façon plus efficace.

Qu'est-ce qu'une DTD et à quoi sert-elle ?
Une DTD (document type definition) définit la syntaxe valide pour une classe de documents XML. C'est à dire qu'elle dresse une liste des noms d'éléments, définit les éléments pouvant apparaître ensemble, détermine les attributs disponibles pour chaque type d'élément, etc. Une DTD utilise une syntaxe différente de celle des documents XML.

Les développeurs Web doivent-ils avoir recours à une DTD lorsqu'ils utilisent XML pour décrire des données ?
Non. XML peut être utilisé pour décrire les données avec ou sans DTD. On parle de XML "valide" lorsque l'on fait référence à des données XML qui référencient une DTD, tandis que l'on parle de XML "correctement formulé" lorsqu'on fait référence à des données XML qui n'utilisent pas de DTD. L'ajout du XML "correctement formulé" est l'une des différences fondamentales entre XML et SGML (Standard Generalized Markup Language). En clair, dans les deux cas, XML doit être conforme aux normes du langage (par exemple, toutes les balises doivent être fermées et elles ne peuvent pas se chevaucher).

Que sont les schémas XML ? En quoi diffèrent-ils des DTD ?
Si XML 1.0 fournit un mécanisme (la DTD) pour définir le modèle de contenu d'un document XML, il est évident qu'une méthode plus complète et rigoureuse de définition d'un modèle de contenu est nécessaire. Un schéma XML est la définition (à la fois en terme d'organisation et en termes de types de données) d'une structure XML spécifique. Un schéma XML utilise le langage XML Schema pour définir chaque type d'élément du schéma et déterminer le type de données que l'élément lui a associé. L'une des fonctionnalités les plus séduisantes des schémas par rapport aux DTD tient probablement au fait qu'un schéma est également un document XML. Cela signifie qu'il peut être lu par les mêmes outils que les documents XML qu'il décrit.

Les services XML de Microsoft prennent actuellement en charge les schémas de données XML, ce qui témoigne du travail sur les schémas effectué par W3C pour le lancement d'Internet Explorer 5 en mars 1999. Les schémas de données XML permettent aux développeurs d'ajouter des types de données à leurs documents XML et de définir les modèles de contenu ouverts. Cette extension des fonctionnalités des DTD sont un atout essentiel de la programmation avec XML.

Toutefois, le W3C prépare des XSD (XML Schema Definitions) qui constitueront la norme des schémas XML. Microsoft prévoit d'intégrer à ses services XML principaux une prise en charge des XSD dès que les spécifications seront officiellement des recommandations.

Que sont les espaces de noms et en quoi sont-ils si importants ?
Les espaces de noms sont une autre fonctionnalité avancée de XML et figurent dans une note du W3C comme partie intégrante des spécifications de XML 1.0. Ils permettent aux développeurs de qualifier les noms d'éléments et leurs relations. Les espaces de noms rendent les noms d'éléments identifiables de façon unique, permettant ainsi d'éviter les conflits entre des éléments ayant un nom identique mais définis dans des vocabulaires différents. Ils permettent de mélanger les balises provenant de divers espaces de noms, ce qui est indispensable si les données proviennent de plusieurs sources.

Par exemple, une librairie peut définir la balise <TITLE> pour représenter le titre d'un livre, dans l'élément <BOOK> uniquement. En revanche, dans un annuaire, la balise <TITLE> peut indiquer la position hiérarchique d'une personne, par exemple :

<TITLE>President</TITLE>
Les espaces de noms permettent de faire clairement cette distinction.

(concernant XSLT et Xpath)
Qu'est-ce que XSLT ?
XSLT (Extensible Stylesheet Language for Transformations) est une recommandation officielle du W3C approuvée le 16 novembre 1999. Il s'agit d'un langage, tant au sens de marquage que de programmation, dans la mesure ou il fournit un mécanisme de transformation d'une structure XML en une autre structure XML, en HTML ou en tout autre format texte (tel que SQL). Si XSLT peut être utilisé pour créer l'affichage d'une page Web, sa véritable utilité réside dans sa capacité à modifier les structures sous-jacentes et pas seulement les représentations de leur support, comme dans le cas des feuilles de style en cascade (CSS).

Quelle est la différence entre XSL, XQL, les modèles XSL et XSLT ?
XSLT a été développé en raison des limites des feuilles CSS qui ne permettaient pas d'apporter des modifications structurelles à un document XML, à une époque où la raison essentielle de la création de XML était davantage de trouver un substitut à HTML que de fournir un langage universel de description des données. Avec le langage XSL (Extensible Stylesheet Language) on a par conséquent cherché une nouvelle façon de mettre en forme XML.

Toutefois, il est devenu rapidement évident, à la fois pour les participants au groupe de travail sur les styles du W3C et pour les premiers partisans de XML, qu'un langage qui pourrait transformer XML d'un format à un autre simplifierait radicalement une grande partie du code généré. Microsoft a remis une proposition au W3C, initialement appelée XQL (XML Query Language), qui a été adoptée par le W3C comme langage de modèle XSL. La plupart des fonctionnalités de ce langage ont été intégrées dans les spécifications XSLT finales.

La norme qui en a résulté intègre des paramètres permettant d'adapter XSLT à des conditions initiales multiples, ainsi que des modèles nommés pour créer des blocs fonctionnels de code et plusieurs fonctions améliorées pour la manipulation numérique et la manipulation des chaînes. XSLT permet également d'ajouter des fonctionnalités incorporées dans le langage, atout dont Microsoft tire parti dans son implémentation pour ajouter plusieurs fonctionnalités très utiles, notamment l'accès au objets COM et l'utilisation du script.

Qu'est ce que XPath ?
XPath est un langage de requête défini pour XML qui fournit une syntaxe simple permettant de sélectionner le sous-ensemble de nî uds d'un document. Avec XPath, vous pouvez rechercher des collections d'éléments en spécifiant un chemin d'accès de type répertoire (donc un nom) ainsi que les conditions relatives à ce chemin d'accès. XPath est essentiel pour XSLT et pour le DOM XML et a également des liens avec les spécifications de XPointer (vous permettant de sélectionner des fragments de documents en fonction d'une combinaison d'URL [Uniform Resource Locators] et d'expressions XPath).

Pourquoi XSLT est-il si important pour XML ?
XSLT est un langage qui transforme un document XML en un autre document, c'est-à-dire qui fournit un mécanisme permettant d'utiliser comme source unique des données XML, de créer des affichages élaborés pour les pages Web que l'utilisateur peut modifier dynamiquement et de filtrer les données pour des communications spécifiques. XSLT est suffisamment fiable pour coder les règles métier. Il peut générer des graphiques (outre les pages Web) à partir des données. Il peut même gérer les communications avec d'autres serveurs (notamment en association avec les modules de script qui peuvent être intégrés dans le XSLT) et il est capable de produire les messages appropriés au sein même de XSLT. S'il est peu probable qu'il remplace les nombreuses interactions à l'intérieur des systèmes de bureau (pour des raisons de performances et de facilité d'utilisation), XSLT devrait devenir dans les prochaines années l'un des principaux langages de "programmation" pour la communication entre systèmes.

Quelle est la différence entre XSLT et CSS ? Ne s'agit-il pas dans les deux cas de feuilles de style ?
Les feuilles de style en cascade (CSS) fonctionnent en attribuant un ensemble de propriétés d'affichage à un élément HTML. Les feuilles CSS déterminent l'apparence visuelle d'une page mais ne modifient pas la structure du document source.

Quant à XSLT, il s'agit d'un langage basé sur un modèle qui vous permet de mapper un certain modèle dans le document source et d'écrire le résultat en XML, en HTML ou en texte brut. Avec XSLT, vous pouvez transformer la structure d'un document XML en un document XML différent. Par exemple, vous pouvez changer l'ordre d'un document XML, ajouter ou supprimer des éléments, réaliser des tests conditionnels ou effectuer des itérations dans des collections d'éléments.

XSLT et CSS ne sont pas des normes incompatibles. Pour créer des pages Web en XML, il existe une technique pratique consistant à utiliser XSLT pour transformer XML en des structures telles que des listes ou des tables, puis à appliquer les feuilles CSS au résultat de façon à contrôler l'apparence de ces structures sur le support approprié. Vous pouvez même créer des feuilles CSS à partir de XSLT.

(Normes)
Dans quelle mesure les normes XML sont-elles respectées par Microsoft ?
Microsoft a été l'un des premiers à s'intéresser à XML, pratiquement dès l'apparition du langage, et effectivement, la plupart des recommandations et des ébauches de travail produites par le W3C dans les dernières années ont nécessité l'intervention d'un voire de plusieurs employés de Microsoft. Microsoft s'est fortement engagé à collaborer avec les organismes de normalisation du W3C de façon à garantir que XML se développe d'une façon bénéfique pour les utilisateurs, et l'entreprise a considérablement contribué au développement de divers domaines, notamment les spécifications XML, DOM, XSLT et les langages de définition des schémas. Microsoft s'engage à se conformer aux normes et spécifications qui seront élaborées.

Quelle est la relation entre XML et le World Wide Web Consortium (W3C) ?
Le W3C a un groupe de travail actif sur XML. Microsoft a été l'un des co-fondateurs de ce groupe en juin 1996 et depuis lors, de nombreux acteurs industriels l'ont rejoint, notamment Netscape Communications Corp., IBM et Oracle. Pour plus d'informations sur le processus de normalisation de XML, consultez le site Web du W3C (en anglais).

Quel est le statut de XML pour le W3C ?
La version 1.0 de XML a été officiellement ratifiée en décembre 1998 et constitue désormais une norme stable. Pour plus d'informations sur les spécifications actuelles de XML et sur le processus de soumission et d'évaluation mené par le W3C, veuillez consulter le site Web du W3C (en anglais).

Quel est le statut de DOM pour le W3C ?
Le document du W3C sur DOM Niveau 1 a un statut de Recommandation. Cela signifie que le W3C le recommande désormais comme une norme sur le World Wide Web. Pour plus d'informations sur le DOM et sur le processus de soumission et d'évaluation au sein du W3C, veuillez consulter les spécifications DOM (en anglais) .

(Prise en charge des outils)
SQL Server et ADO prennent-ils en charge XML ?
La technologie Microsoft ADO (ActiveX Data Objects) fournit plusieurs méthodes de conversion des jeux d'enregistrements de bases de données (collections d'enregistrements de données) en un format XML, ainsi que des outils permettant de prendre XML dans une structure données et de le renvoyer dans n'importe quelle base de données prise en charge par ADO (y compris les bases de donnés SQL Server et Oracle). De plus, grâce à l'objet source de données XML en MSXML2 et MSXML3, vous pouvez choisir de charger XML directement dans ADO pour générer des jeux d'enregistrements.

SQL Server 2000 vous permet de définir et de récupérer directement XML à partir d'une URL de la même façon que pour appeler une page Web. Il s'agit là d'un mécanisme particulièrement efficace de gestion des données car il vous permet d'intégrer vos données SQL Server dans des filtres XSL, à l'intérieur des pages Web

et à n'importe quel emplacement pouvant accueillir un document XML. De plus, vous pouvez installer des modèles personnalisés pour contrôler la façon dont XML est produit à partir des données SQL Server et faire de la base de données un outil efficace de production de pages XHTML.

Enfin, les applications telles que BizTalk Server vous permettent d'effectuer des mappages entre diverses sources de données (qu'il s'agisse de documents XML, de bases de données ou de documents Excel ou Word), de créer des pipelines de données complexes pour votre architecture Web et de créer des schémas efficaces pour satisfaire les besoins de votre base de données XML.

Existe-t-il actuellement des outils Microsoft susceptibles de m'aider à utiliser XML rapidement ?
Microsoft BizTalk Server 2000, serveur XML d'échanges mutuels de données, fournit l'infrastructure et les outils permettant aux entreprises de commerce électronique de fonctionner. BizTalk Server repose sur une infrastructure réglementée de routage, de transformation et de suivi des documents commerciaux. Cette infrastructure permet aux entreprises d'intégrer, de gérer et d'automatiser les processus commerciaux en échangeant des documents commerciaux (par exemple des commandes et des factures) entre applications au sein d'une entreprise ou entre plusieurs organisations. Pour plus d'informations, veuillez consulter le site Microsoft BizTalk Server 2000 (en anglais).

Qu'est-ce que SOAP ?
Le protocole SOAP (Simple Object Access Protocol) permet de créer des environnements informatiques complexes et largement distribués fonctionnant sur Internet en utilisant l'infrastructure Internet existante. SOAP est un outil performant permettant aux applications de communiquer entre elles sur Internet. Pour plus d'informations sur SOAP, veuillez consulter les spécifications SOAP (en anglais).

(Problèmes et solutions)
Pourquoi mon objet document reste-t-il vide après que j'ai appelé la méthode Load() ?
Par défaut, les opérations sont chargées de façon asynchrone. Cela signifie que si vous fournissez une adresse URL http, la méthode load() sera immédiatement renvoyée et votre objet de document restera vide car les données ne seront pas encore revenues du serveur. Pour éviter cela, ajoutez la ligne suivante à votre code :

xmldoc.async = false;
De plus, si vous chargez des document XML http à partir d'une application C++ autonome, vous devrez interroger la file d'attente des messages pour continuer le téléchargement.

Comment charger un document contenant des caractères étrangers et spéciaux ?
Un document peut contenir des caractères étrangers tels que les caractères suivants :

<test>foreign characters (úóíá) </test>
Les caractères étrangers comme úóíá doivent être précédés d'une séquence d'échappement. Les caractères étrangers peuvent être codés en UTF-8 ou spécifiés via un codage différent de la manière suivante :
<?xml version="1.0" encoding="iso-8859-1"?>
<test>foreign characters (úóíá) </test>
Désormais, votre XML se chargera correctement.

D'autres caractères sont réservés dans XML et doivent également être gérés différemment. Le XML suivant :

<foo>This & that</foo>
génère l'erreur suivante :
Aucun espace blanc n'est autorisé à cet emplacement.
Line 0000001: <foo>This & that</foo>
Pos 0000012: ----------^
Le caractère perluète (&) fait partie de la structure syntaxique de XML et ne sera pas interprété comme un caractère perluète s'il est simplement placé dans la source de données XML. Vous devez lui substituer une séquence de caractères spéciaux appelés "entité".

Les caractères suivants requièrent les entités correspondantes suivantes :

< <
& &
> >
" "
' '

« Dans le langage HTML, les caractères « inférieur à » (<), « supérieur à » (>) et la perluète (&), « ampersand » en anglais, sont utilisés dans des commandes HTML. Lorsqu'on veut s'en servir dans le texte d'un document HTML, il faut avoir recours à leur code alphabétique HTML ou à leur code numérique HTML respectif. »(http://www.olf.gouv.qc.ca/ressources/internet/fiches/2075148.htm)
Le caractère esperluette (&) est utilisé essentiellement pour faire appel à une référence d'entité soit prédéfinie, soit générale, déclarée au préalable dans la DTD.
&
&entité_générale;
Les caractères inférieur à (<), l'esperluette (&) ou la séquence]]> doivent être remplacés respectivement par les références d'entité <, & ou]]>

Exemples :
```
<?xml version="1.0" encoding="UTF-8"?>
<mon-document>
    Ceci est un document
écrit en français.
</mon-document>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<mon-document>
    Ceci est un document
&#233;crit en fran&#231;ais.
</mon-document>
```

En utilisant des entités définies dans une DTD (interne ou externe):
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mon-document [
    <!ENTITY e-accent-aigu "&#233;">
    <!ENTITY c-cedille "&#231;">
]>
<mon-document>
    Ceci est un document
&e-accent-aigu;crit en fran&c-cedille;ais.
</mon-document>
```

Comment utiliser les entités HTML dans mon XML ?
Le XML suivant contient une entité HTML :

```
<copyright>Copyright © 1999, Microsoft Inc, All rights reserved.</copyright>
```
Il génère l'erreur suivante :

```
Référence à l'entité non définie 'copy'.
Line: 1, Position: 23, ErrorCode: 0xC00CE002
<copyright>Copyright © 1999, ...
----------------------^
```

En effet, XML n'a que cinq entités intégrées. Pour plus d'informations sur les entités intégrées, veuillez vous reporter à la section Comment charger un document contenant des caractères spéciaux ?.

Pour utiliser les entités HTML, vous devez les définir à l'aide d'une DTD. Pour plus d'informations sur les DTD, veuillez consulter les Recommandations du WC3 sur XML (en anglais). Pour utiliser cette DTD, insérez-la directement dans une balise DOCTYPE de la manière suivante :
<!DOCTYPE foo SYSTEM "http://msdn.microsoft.com/xml/general/htmlentities.dtd">
<copyright>Copyright © 1999, Microsoft Inc, All rights reserved.</copyright>
Pour que le chargement puisse s'effectuer, vous devez désactiver la propriété validateOnParse de l'interface IXMLDOMDocument. Essayez de coller cela dans la page de test du valideur, désactivez la validation DTD et cliquez sur Valider. Remarquez que le document se charge et que le caractère de copyright est disponible dans l'arborescence du DOM apparaissant à la fin de la page du valideur.

Si vous effectuez déjà la validation DTD, vous devez alors inclure les entités HTML comme une entité de paramètre dans votre DTD de la manière suivante :

<!ENTITY % HTMLENT SYSTEM "http://msdn.microsoft.com/xml/general/htmlentities.dtd">
%HTMLENT;
Cette opération permettra de définir toutes les entités HTML pour que vous puissiez les utiliser dans votre document XML.

Comment les espaces blancs sont-ils gérés dans le contenu d'un élément ?
Le DOM XML utilise trois méthodes pour accéder au contenu textuel des éléments :

Propriétés Comportements
nodeValue Renvoie le contenu du texte original (y compris les espaces) sur les nî uds TEXT, CDATA, COMMENT et PI comme spécifié dans la source XML d'origine. Renvoie "null" sur les nî uds ELEMENT et sur le DOCUMENT.
data Même comportement que nodeValue
text Procède à la concaténation récursive des divers nî uds TEXT et CDATA dans un sous-arbre donné et renvoie le résultat combiné.

Remarque : Les espaces blancs correspondent aux caractères de nouvelle ligne, de tabulation et d'espace.

La propriété nodeValue renvoie toujours les données contenues dans le document original, indépendamment de la façon dont le document est chargé et de la portée de xml:space.

La propriété text concatène tout le texte présent dans le sous-arbre spécifié et étend les entités. Cette opération dépend de la façon dont le document est chargé, de l'état du commutateur preserveWhiteSpace et de la portée de xml:space, comme illustré ci-dessous :

preserveWhiteSpace = true lorsque le document est chargé preserveWhiteSpace=true preserveWhiteSpace=true preserveWhiteSpace=false preserveWhiteSpace=false
xml:space=preserve xml:space=default xml:space=preserve xml:space=default
preserved preserved preserved preserved and trimmed

preserveWhiteSpace = false lorsque le document est chargé preserveWhiteSpace=true preserveWhiteSpace=true preserveWhiteSpace=false preserveWhiteSpace=false
xml:space=preserve xml:space=default xml:space=preserve xml:space=default
half preserved half preserved and trimmed half preserved half preserved and trimmed

"Preserved" signifie que le contenu exact du texte d'origine est conservé tel qu'il se présente dans le document XML original, "trimmed" signifie que les espaces à gauche et à droite sont éliminés et "half preserved" signifie que les "espaces blancs significatifs" sont préservés tandis que les "espaces blancs non significatifs" sont

normalisés. Les espaces blancs significatifs sont ceux contenus dans le texte. Les espaces blancs non significatifs sont ceux compris entre des balises comme dans l'exemple suivant :

<name>\n
\t<first> Jane</first>\n
\t<last>Smith </last>\n
</name>
Dans cet exemple, la couleur rouge est un espace blanc non significatif et peut être ignorée, tandis que la couleur verte est un espace blanc significatif puisqu'elle fait partie du contenu du texte et a par conséquent une signification qui ne peut être ignorée. Dans cet exemple, la propriété text renvoie donc les résultats suivants :

états valeurs renvoyées
preserved "\n\t Jane\n\tSmith \n"

preserved and trimmed "Jane\n\tSmith"

half preserved " Jane Smith "

half preserved and trimmed "Jane Smith"

Remarquez que l'état "half preserved" normalise les espaces blancs non significatifs. Ainsi, les caractères de nouvelle ligne et de tabulation sont condensés en un seul caractère d'espace. Vous pouvez modifier les attributs xml:space et le commutateur preserveWhiteSpace ; la propriété text renverra alors une valeur différente tenant compte de ces changements.

CDATA et les limites du sous-arbre xml:space="preserve"
Dans l'exemple suivant, le contenu du nî ud CDATA ou nî ud "preserved" est concaténé tel quel et ne participe pas à la normalisation des espaces blancs non significatifs. Considérons l'exemple suivant :

<name>\n
\t<first> Jane </first>\n
\t<last><![CDATA[Smith]></last>\n
</name>
Dans ce cas, les espaces blancs dans le nî ud CDATA ne sont jamais "fusionnés" avec les espaces blancs "non significatifs" et ne sont jamais rognés. Par conséquent, dans le cas d'un nî ud "half preserved and trimmed", le résultat renvoyé sera le suivant :

"Jane Smith "
Ici, l'espace blanc non significatif entre les balises </first> et <last> est inclu quel que soit le contenu du nî ud CDATA. Le même résultat est renvoyé si le CDATA est remplacé de la façon suivante :

<last xml:space="preserve"> Smith </last>
Les entités sont spéciales

Les entités sont chargées et analysées comme partie intégrante de la DTD et apparaissent sous le nî ud DOCTYPE. Elles n'ont pas nécessairement une portée xml:space. Par exemple :

<!DOCTYPE foo [
<!ENTITY Jane "<employee>\n
\t<name> Jane </name>\n
\t<title>Software Design Engineer</title>\n
</employee>">
]>
<foo xml:space="preserve">&Jane;</foo>

En admettant que preserveWhiteSpace=false (dans le champ d'application de la balise DOCTYPE), les espaces blancs non significatifs sont perdus lorsque l'entité est analysée. L'entité n'aura pas de nî uds d'espaces blancs. L'arborescence se présentera de la façon suivante :

DOCTYPE foo
ENTITY: Jane
ELEMENT: employee
ELEMENT: name
TEXT: Jane
ELEMENT: title
TEXT>:Software Design Engineer
ELEMENT: foo
ATTRIBUTE: xml:space="preserve"
ENTITYREF: Jane

Remarquez que l'arborescence DOM exposée sous le nî ud ENTITY dans le DOCTYPE ne contient pas de nî ud WHITESPACE. Cela signifie que les enfants du nî ud ENTITYREF n'auront pas non plus de nî ud WHITESPACE même si la référence à l'entité est dans le champ d'application de xml:space="preserve".

Chaque instance d'un nî ud ENTITY référencée dans un document donnée produit toujours la même arborescence.

Si une entité doit absolument préserver les espaces blancs, elle doit spécifier son propre attribut xml:space ou le commutateur preserveWhiteSpace du document doit être défini sur true.

Comment les espaces blancs sont-ils gérés dans les attributs ?
Il existe plusieurs façons d'accéder à la valeur d'un attribut. L'interface IXMLDOMAttribute a une propriété nodeValue qui correspond à nodeValue et une propriété text qui est l'extension Microsoft. Ces propriétés renvoient le résultat suivant : propriétés texte renvoyé
attrNode.nodeValue
attrNode.value
getAttribute("name") Renvoie le contenu exact (avec les entités étendues) du document d'origine.
attrNode.nodeTypedValue Null
attrNode.text Même chose qu'avec nodeValue sauf que les espaces blancs à droite et à gauche sont éliminés.

Les spécifications du langage XML définissent le comportement suivant pour les applications XML : Types d'attributs Texte renvoyé
CDATA ID, IDREF, IDREFS, ENTITY, ENTITIES, NOTATION, enumeration
partiellement normalisé entièrement normalisé
"Partiellement normalisé" signifie que les caractères de nouvelle ligne et les tabulations sont converties en espaces mais que les espaces multiples ne sont pas condensés en un seul espace.

Comment les espaces blancs sont-ils gérés dans le modèle d'objet XML ?
Parfois, le modèle d'objet XML affiche des nî uds TEXT contenant des espaces blancs. Cela peut générer une confusion, les espaces blancs étant généralement éliminés. Prenons l'exemple XML suivant :

```
<?xml version="1.0" ?>
<!DOCTYPE person [
 <!ELEMENT person (#PCDATA|lastname|firstname)>
 <!ELEMENT lastname (#PCDATA)>
 <!ELEMENT firstname (#PCDATA)>
]>
<person>
 <lastname>Smith</lastname>
```

<firstname>John</firstname>
</person>

On obtient l'arborescence suivante :

Processing Instruction: xml
DocType: person
ELEMENT: person
TEXT:
ELEMENT: lastname
TEXT:
ELEMENT: firstname
TEXT:

Le prénom (first name) et le nom de famille (last name) sont entourés de nî uds TEXT contenant uniquement un espace blanc parce que le modèle de contenu pour l'élément "person" est MIXTE ; il contient le mot clé #PCDATA. Un modèle de contenu MIXTE indique que du texte peut être intercalé entre les éléments. Par conséquent, les lignes suivantes sont également valides :

<person>
My last name is <lastname>Smith</lastname> and my first name is
<firstname>John</firstname>
</person>

et l'on obtient une arborescence similaire à la suivante :

ELEMENT: person
TEXT: My last name is
ELEMENT: lastname
TEXT: and my first name is
ELEMENT: firstname
TEXT:

Sans l'espace blanc après le mot "is" et avant <lastname>, et l'espace blanc après </lastname> et avant le mot "and", la phrase serait incompréhensible. Par conséquent, pour des modèles de contenu MIXTE, le fait de combiner texte, espaces blancs et éléments est pertinente, ce qui n'est pas le cas avec les modèles de documents non-MIXTE.

Pour faire disparaître les nî uds TEXT comprenant uniquement des espaces blancs, supprimez le mot-clé #PCDATA de la déclaration de l'élément "person" :

<!ELEMENT person (lastname,firstname)>
ce qui produit une arborescence claire :

Processing Instruction: xml
DocType: person
ELEMENT: person
ELEMENT: lastname
ELEMENT: firstname

Quelle est la fonction de la déclaration XML ?
La déclaration XML doit apparaître au début du document XML :

<?xml version="1.0" encoding="utf-8"?>

Elle spécifie les éléments suivants :

Il s'agit d'un document XML. Il peut être utilisé par les renifleurs MIME pour détecter qu'un fichier est de type text/xml lorsque le type MIME a été perdu ou n'a pas été spécifié.
Le document suit les spécifications de XML 1.0. Cela s'avérera important à l'avenir, lorsque de nouvelles versions de XML seront diffusées.
Codage des caractères du document. L'attribut de codage est optionnel et défini par défaut sur UTF-8.
Remarque : La déclaration XML doit être en première ligne dans un document XML ; par conséquent, le fichier XML suivant :

<!--HEADLINE="Dow closes as techs get hammered"-->
<?xml version="1.0"?>
génère l'erreur d'analyse suivante :

Déclaration xml non valide.
Line 0000002: <?xml version="1.0"?>
Pos 0000007: ------^
Remarque : La déclaration XML est optionnelle. Si vous avez besoin de spécifier un commentaire ou de traiter une instruction au début, n'insérez aucune déclaration XML. Toutefois, le codage sera défini par défaut sur UTF-8.

Comment imprimer un document XML dans un format lisible ?
Lorsque vous générez un fichier XML en créant entièrement un document à partir du DOM, toutes les informations se retrouvent sur une seule ligne, sans espaces blancs entre elles. Il s'agit là du comportement par défaut.

La feuille de style XSL par défaut créée dans Internet Explorer 5 affiche et imprime les documents XML dans un format lisible. Par exemple, si Internet Explorer 5 est installé sur votre machine, essayez d'afficher le fichier nospace.xml. Vous devriez voir apparaître dans votre navigateur l'arborescence suivante :

- <ORDER>
 - <ITEM NAME="123">
 <NAME>XYZ</NAME>
 <PRICE>12.56</PRICE>
 </ITEM>
 </ORDER>
Aucun espace blanc n'est inséré dans le XML.

Imprimer XML de façon lisible n'est pas chose facile, surtout si vous avez une DTD définissant différents types de modèles de contenu. Ainsi, dans le modèle de contenu mixte (#PCDATA), il sera peut-être préférable de ne pas insérer d'espace car cela pourrait changer la signification du contenu. Par exemple, considérons le XML suivant :

<B>E</B><I>lephant</I>
Il ne faut pas que cela devienne :

<B>E</B>
<I>lephant</I>
car les limites du mot ne serait plus correctes.

Tout cela rend l'impression automatique problématique. Pour imprimer XML de façon lisible, vous pouvez utiliser le DOM afin d'insérer des espaces blancs comme des nî uds de texte aux endroits appropriés.

Comment utiliser les espaces de noms dans les DTD ?

Pour utiliser un espace de noms dans une DTD, il vous faut le déclarer dans la déclaration ATTLIST de l'élément qui l'utilise en procédant de la manière suivante :

```
<!ELEMENT x:customer ANY >
<!ATTLIST x:customer xmlns:x CDATA #FIXED "urn:...">
```

L'espace de noms doit être de type #FIXED. Les espaces de noms sur les attributs fonctionnent de la même façon :

```
<!ELEMENT customer ANY >
<!ATTLIST customer
 x:value CDATA #IMPLIED
 xmlns:x CDATA #FIXED "urn:...">
```

Espaces de noms et schémas XML
Les DTD et les schémas XML ne peuvent pas être mélangés. Par exemple,

```
xmlns:x CDATA #FIXED "x-schema:myschema.xml"
```

n'entraînera pas l'utilisation des définitions de schéma spécifiées dans myschema.xml. Les DTD et les schémas XML sont incompatibles.

Comment utiliser XMLDSO dans Visual Basic ?
Prenons pour exemple le XML suivant :

```
<contacts>
 <person>
 <name>Mark Hanson</name>
 <telephone>206 765 4583</telephone>
 </person>
 <person>
 <name>Jane Smith</name>
 <telephone>425 808 1111</telephone>
 </person>
</contacts>
```

Vous pouvez créer un lien vers des jeux d'enregistrements ADO de la manière suivante :

Créez un nouveau projet VB 6.0.
Ajoutez des références à Microsoft ActiveX Data Objects, version 2.1 ou ultérieure, à Microsoft Data Adapter Library et à Microsoft XML, version 2.0.
Chargez les données XML dans le contrôle DSO XML en utilisant le code suivant :

```
Dim dso As New XMLDSOControl
Dim doc As IXMLDOMDocument
Set doc = dso.XMLDocument
doc.Load ("d:\test.xml")
```

Mappez le DSO dans un nouvel objet Recordset en utilisant un DataAdapter avec le code suivant :

```
Dim da As New DataAdapter
Set da.Object = dso
Dim rs As New ADODB.Recordset
Set rs.DataSource = da
```

Accédez aux données :

```
MsgBox rs.Fields("name").Value
```

Cela permet d'afficher a chaîne "Mark Hanson"
Comment utiliser le DOM XML avec Java ?
La version Internet Explorer 5 de MSXML.DLL doit déjà être installée. Avec Visual J++ 6.0, sélectionnez dans le menu Projet Ajouter un wrapper COM, puis choisissez Microsoft XML 1.0 dans la liste d'objets COM. Cela

permet de créer les wrappers Java requis dans un nouveau lot appelé "msxml s;». Ces wrappers Java pré-créés peuvent également être téléchargés. Les classes peuvent être utilisées de la façon suivante :

```
import com.ms.com.*;
import msxml.*;
public class Class1
{
 public static void main (String[] args)
 {
 DOMDocument doc = new DOMDocument();
 doc.load(new Variant("file://d:/samples/ot.xml"));
 System.out.println("Loaded " + doc.getDocumentElement().getNodeName());
 }
}
```

L'exemple de code charge un fichier de test "ot.xml" de 3,8 Mo à partir de l'exemple Sun. La classe Variant est utilisée pour encapsuler le type primaire VARIANT Win32.

Vous ne pouvez pas utiliser les comparaisons de pointeurs sur les nî uds puisque chaque fois que vous récupérez un nî ud, vous obtenez en fait un nouveau wrapper. Donc plutôt que d'utiliser le code suivant,

```
IXMLDOMNode root1 = doc.getDocumentElement();
IXMLDOMNode root2 = doc.getDocumentElement();
if (root1 == root2)...
```

utilisez celui-ci :

```
if (ComLib.isEqualUnknown(root1, root2)) ....
```

La taille totale des wrappers .class est d'environ 160 Ko. Toutefois, si vous souhaitez vous conformer pleinement aux spécifications du W3C, vous devrez utiliser uniquement les wrappers IXMLDOM*. Les classes suivantes sont les anciennes interfaces XML de IE 4.0 et peuvent être supprimées du dossier msxml :

```
IXMLAttribute*,
IXMLDocument*, XMLDocument*
IXMLElement*,
IXMLError*,
IXMLElementCollection*,
tagXMLEMEM_TYPE*
_xml_error*
```

La taille peut ainsi être réduite à 147 Ko. Vous pourrez également supprimer les éléments suivants :

DOMFreeThreadedDocument
Accède au document XML à partir de plusieurs threads dans une application Java.
XMLHttpRequest
Communique avec les serveurs en utilisant les extensions HTTP DAV XML.
IXTLRuntime
Définit l'objet de script de la feuille de style XSL.
XMLDSOControl
Crée un lien vers les données XML dans une page HTML.
XMLDOMDocumentEvents
Renvoie des rappels lors de l'analyse.
Cela permet de réduire la taille à 116 Ko. Pour réduire encore le nombre d'octets, pensez que le DOM lui-même est composé de deux couches : une couche principale comprenant :

```
DOMDocument, IXMLDOMDocument
IXMLDOMNode*
IXMLDOMNodeList*
```

IXMLDOMNamedNodeMap*
IXMLDOMDocumentFragment*
IXMLDOMImplementation
IXMLDOMParseError
et des informations DTD que vous souhaiterez probablement conserver :

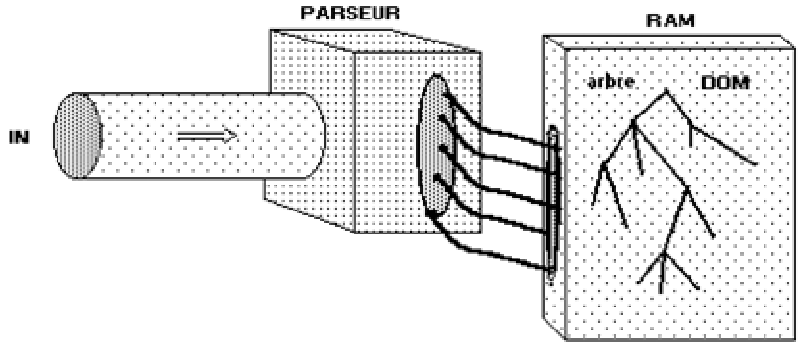
IXMLDOMDocumentType
IXMLDOMEntity
IXMLDOMNotation
Tous les nî uds d'un document XML sont de type IXMLDOMNode, ce qui offre une fonctionnalité complète, mais des wrappers de niveau supérieur existent pour chaque type de nî ud. Par conséquent, toutes les interfaces suivantes peuvent également être supprimées si vous modifiez le wrapper DOMDocument et changez ces types spécifiques pour utiliser IXMLDOMNode :

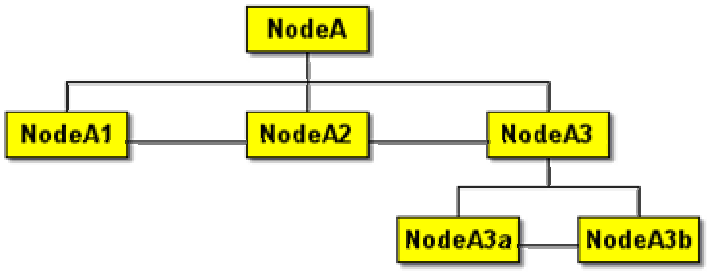
IXMLDOMAttribute
IXMLDOMCDATASection
IXMLDOMCharacterData
IXMLDOMComment
IXMLDOMElement
IXMLDOMProcessingInstruction
IXMLDOMEntityReference
IXMLDOMText
Ces suppressions permettent de faire passer le nombre d'octets à 61 Ko. Néanmoins, avec IXMLDOMElement, les méthodes getAttribute et setAttribute sont utiles. Sinon, vous devrez utiliser :

IXMLDOMNode.getAttributes().setNamedItem(...)

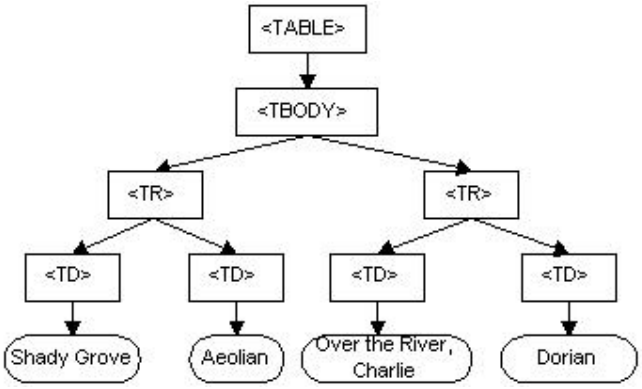
DOM

Différentes visions du « Document Object Model »





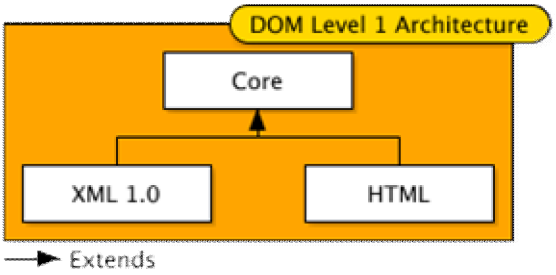
- NodeA.`firstChild` = NodeA1
- NodeA.`lastChild` = NodeA3
- NodeA.`childNodes.length` = `3`
- NodeA.`childNodes[0]` = NodeA1
- NodeA.`childNodes[1]` = NodeA2
- NodeA.`childNodes[2]` = NodeA3
- NodeA1.`parentNode` = NodeA
- NodeA1.`nextSibling` = NodeA2
- NodeA3.`prevSibling` = NodeA2
- NodeA3.`nextSibling` = `null`
- NodeA.`lastChild.firstChild` = NodeA3a
- NodeA3b.`parentNode.parentNode` = NodeA



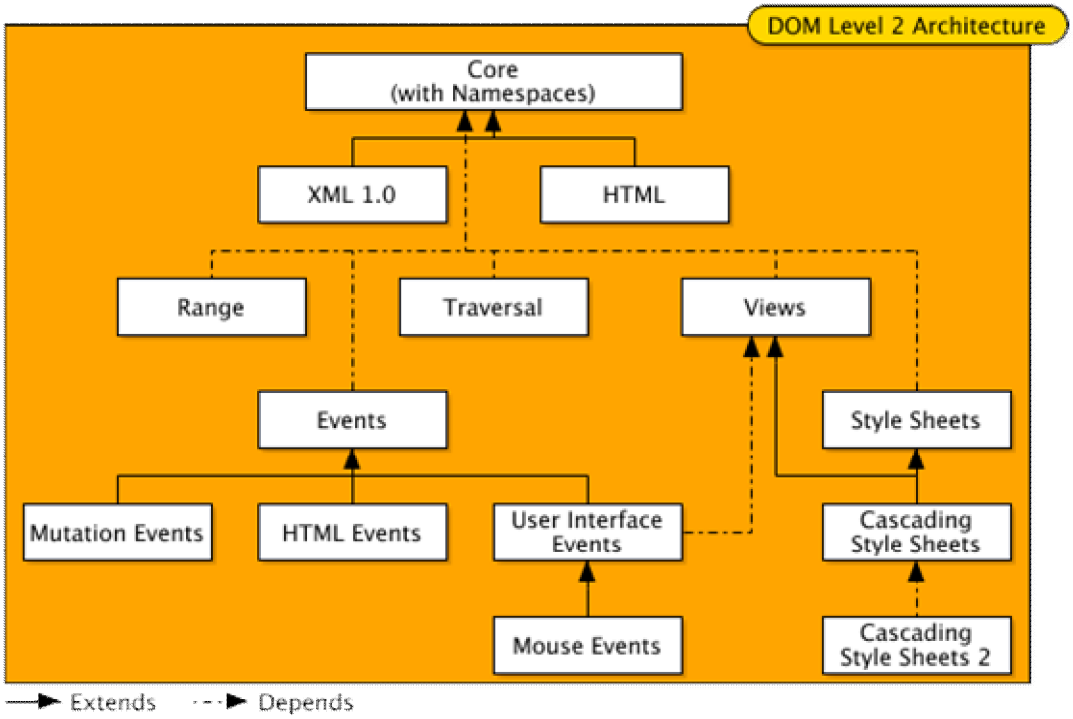
Représentation dûn tableau en HTML

Il existe plusieurs « DOM »

Le premier niveau des specifications est finalisé en Octobre 1998 et supporte XML 1.0 et HTML.



DOM niveau 2
Le deuxième niveau des spécifications est achevé depuis Novembre 2000.
Celui-ci étends le niveau 1 en supportant XML 1.0 avec les espaces de noms et le support des feuilles de style (CSS), les évenements et la manipulation sous forme dôarbre.



Le niveau 3 est en développementé

VALIDEUR TEMPS REEL

(Source à enregistrer avec l'extension .htm)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- saved from url=(0052)http://www.architag.com/xmlu/play/editor/XMLEdit.htm -->
<HTML><HEAD><TITLE>Architag Multi-window Real-time XML Editor</TITLE>
<META content="text/html; charset=windows-1252" http-equiv=Content-Type>
<STYLE>BODY {
	FONT-FAMILY: verdana
}
.errormsg {
	COLOR: red; FONT-SIZE: 7pt; FONT-WEIGHT: bold
}
.errorsrc {
	FONT-FAMILY: monospace; FONT-SIZE: 7pt; MARGIN-TOP: 0px
}
.validind {
	BACKGROUND-COLOR: yellow; BORDER-BOTTOM: black thin solid; BORDER-LEFT: black thin solid; BORDER-RIGHT: black
thin solid; BORDER-TOP: black thin solid; HEIGHT: 10px; WIDTH: 10px
}
</STYLE>

<SCRIPT>
function InitializeXML()
   {
   xmldoc = new ActiveXObject("Microsoft.XMLDOM");
   xmldoc.onreadystatechange = ReadyStateHandler;
   xmldoc.validateOnParse = true;
   }

function Validate()
   {
   errormsg.innerHTML = "";
   errorsrc.innerText = "";
   if (XMLSource.value == "")
      {
      validind.style.backgroundColor = "yellow";
      return;
      }
   xmldoc.loadXML(XMLSource.value);
   }

function ReadyStateHandler()
   {
   if (xmldoc.readyState == 4) //READY_STATE_COMPLETED
      if (xmldoc.parseError.errorCode != 0)
         HandleError();
      else
         validind.style.backgroundColor = "green";
   }

function HandleError()
   {
   var Error = xmldoc.parseError;
   errormsg.innerHTML = "<DIV>" + Error.reason  +
   "</DIV><DIV>Line: " + Error.line + ", Character: " +
   Error.linepos + "</DIV>";

   var code = "";
   if (Error.linepos > 0 && Error.srcText != "")
      {
      code = Error.srcText.replace(/\t/g," ") + "\n";
      for (i = 1; i < Error.linepos; i++)
         code += "-";
      code += "^";
      }
   errorsrc.innerText = code;
   validind.style.backgroundColor = "red";
   }

WidthFactor = 45
HeightFactor = 25
function ResizeBox()
```

```
    {
    document.all.XMLSource.style.width=document.body.clientWidth - 20;
    document.all.XMLSource.style.height=document.body.clientHeight - 290;
    document.all.XMLSource.style.fontSize=(document.body.clientHeight/HeightFactor) + (document.body.clientWidth/WidthFactor);
    document.all.errormsg.style.fontSize=(document.body.clientHeight/25);
    document.all.errorsrc.style.fontSize=(document.body.clientHeight/25);
    }

function ShowBrowser()
    {
    HTMLWindow = window.open('', 'newWin', 'toolbar=yes,location=no,scrollbars=yes,resizable=yes,width=400,height=280');
    StyleDoc = new ActiveXObject("Microsoft.XMLDOM");
    StyleDoc.loadXML('<?xml version="1.0"?><xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl"><xsl:template
match="/"><HTML><BODY STYLE="font-family:Verdana;"><xsl:apply-templates/></BODY></HTML></xsl:template><xsl:template
match="*[node()]"><DIV STYLE="margin-left:1em"><SPAN STYLE="color:#354b75">&lt;<xsl:node-name/><xsl:apply-templates
select="@*"/>></SPAN><xsl:apply-templates select="node()"/><SPAN STYLE="color:#354b75">&lt;/<xsl:node-
name/>></SPAN></DIV></xsl:template><xsl:template match="@*"><SPAN STYLE="color:navy"> <xsl:node-name/>="<SPAN
STYLE="color:black"><xsl:value-of/></SPAN>"</SPAN></xsl:template><xsl:template match="pi()"><DIV STYLE="margin-left:18pt;
color:maroon">&lt;?<xsl:node-name/><xsl:apply-templates select="@*"/>?></DIV></xsl:template><xsl:template match="text()"><xsl:value-
of/></xsl:template></xsl:stylesheet>');
    Result = xmldoc.transformNode(StyleDoc);
    HTMLWindow.document.write(Result);
    HTMLWindow.document.close();
    HTMLWindow.focus();
    }

function ResizeText(Factor)
    {
    WidthFactor += Factor;
    HeightFactor += Factor;
    ResizeBox();
    }
</SCRIPT>

<META content="MSHTML 5.00.2920.0" name=GENERATOR></HEAD>
<BODY onload=InitializeXML();ResizeBox(); onresize=ResizeBox();>
<H2 style="TEXT-ALIGN: center">Architag Real-time XML Editor <FONT size=2>(<A
href="http://architag.com/editor">http://architag.com/editor</A>)</FONT></H2><TEXTAREA id=XMLSource onfocus=Validate();
onkeyup=Validate(); style="FONT-WEIGHT: bold"></TEXTAREA>

<DIV id=Copyright style="FONT-SIZE: 8pt; FONT-STYLE: italic"><A
href="http://architag.com/copyright.html">Copyright </A>© 1999, <A
href="http://architag.com/">Architag International Corporation</A>, all rights
reserved. <A href="javascript:ResizeText(-5)">[Font+]</A>   <A
href="javascript:ResizeText(5)">[Font-]</A> </DIV><BR>
<TABLE width="100%">
  <TBODY>
  <TR>
    <TD align=left>
      <DIV class=validind id=validind
      style="FONT-SIZE: medium">    </DIV></TD>
    <TD align=right><A href="javascript:ShowBrowser()">Show in
  Browser</A></TD></TR></TBODY></TABLE>
<DIV class=errormsg id=errormsg style="FONT-SIZE: medium"></DIV><XMP class=errorsrc id=errorsrc style="FONT-SIZE:
medium"></XMP></BODY></HTML>
```

Espaces de noms

Un document XML peut utiliser des balises définies dans différentes DTDs (ou, plus récemment, dans différents schémas). Pour utiliser des symboles définis ailleurs, un document XML doit "importer" l'espace de noms correspondant. Cela se fait en ajoutant un attribut approprié à l'élément qui va utiliser les symboles de l'espace de noms importé:

```
<uneBaliseDuDocumentCourant xmlns:Préfixe="UrlDeDtdOuSchemaAImporter">
```

Par exemple, pour utiliser les balises HTML dans un document XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<article xmlns:html="http://www.w3.org/Profiles/XHTML-transitional">
```

Pour éviter les ambiguïtés, les éléments utilisant des balises importées doivent alors être de la forme <Préfixe:NomDeBalise ...>. Par exemple:

```
<html:img html:src="..." />
```

Il est à noter que la syntaxe doit rester conforme aux contraintes imposées par XML, d'où le "/>" pour fermer l'élément vide contenant l'image. De plus, n'importe quel élément d'un document XML peut importer un espace de nom. L'espace de noms sera alors accessible au sein de l'élément en question, y compris dans tous les éléments imbriqués. Ainsi, dans l'exemple ci-dessus, les balises HTML seront accessibles dans la totalité de l'élément **article**.

Simple Object Access Protocol

Qu'est ce que **SOAP** ? Quels sont ses avantages ?

SOAP (Simple Object Access Protocol) définit un protocole permettant des appels de procédures à distances (RPC) s'appuyant principalement sur le protocole HTTP et sur XML, mais aussi SMTP et POP. Il permet ainsi de définir des services WEB. Les paquets de données circulent sous forme de texte structuré au format XML (Extensible Markup Language).

Microsoft a basé sur SOAP sa nouvelle architecture services Web .NET. Le protocole SOAP est une note du Consortium W3C dont Microsoft fait partie, mais qui n'est pas spécifique à Microsoft et Windows. IBM a également participé à l'élaboration de ce protocole. De plus il existe des implémentations Java , et Borland vien déjà d'implémenter SOAP sous Windows dans Delphi 6 et sous Linux avec Kylix.

Le principal avantage de SOAP est qu'il repose sur 2 standards XML (pour la structure des messages) et HTTP (pour le transport) bien qu'il soit utilisable avec d'autres protocoles de transport. Par rapport à tous les autres protocole de RPC, celui-ci présente l'avantage de l'interopérabilité, on est indépendant des plate-forme et des languages de programmation. Le second avantage réside dans le déploiement des applications, principalement dans un contexte multi-sîtes, pour communiquer entre 2 sociétes via internet, c'est souvent mission-impossible d'utiliser autre chose que du HTTP ou du POP/SMTP à cause des Firewalls, car pour les autres protocoles il faut les reconfigurer, avec tous les trous de sécurité que cela peut engendrer, et cela implique souvent de longue négotiation avec les administarteurs réseaux.

http://xml.developpez.com/soap.htm

Bibliographie :

Xml étape par étape de Michael J.Young . Microsoft Press
Atelier XML de Jake Sturm . . Microsoft Press
Login Hors-série Décembre 2001 « La programmation XML de A à Z »