<?XML 1.0?>

BEST PRACTICES

Vincent ISOZ (version 4.0 R4)


Please consider the environment - do you really need to print this document!?

Attention la majorité des exemples de ce PDF ne fonctionnent plus dans les navigateurs modernes sans déposer les exercices sur un serveur voire ne fonctionnent qu'avec **Internet Explorer**!

L'objectif premier de ce document est d'introduire le lecteur non spécialiste (non informaticien de métier) à la puissance combinée du XML et de l'ensemble de la gamme MS Office 2003 (Word, Excel, InfoPath, Visio, Publisher, Messenger (si…si), etc.) sans faire trop appel à la programmation.

Un effort particulier de vulgarisation et de présentation a été fait étant donné que la littérature francophone sur le sujet semble quasi inexistante (du moins lorsque ce document avait été rédigé en début 2004). Nous espérons donc que ce support sera utile à tous ceux qui sont intéressés par les possibilités du mélange des technologies Microsoft (bureautique) et de l'XML.

Sans aller dans les détails, ce document permet à un utilisateur expert de la suite bureautique MS Office 2003 et ayant quelques connaissances du développement web XHTML/CSS de se faire une idée des possibilités et des nouveautés réelles (mis à part SharePoint Portal Server et Live Communication Server) de cette nouvelle version de la suite bureautique et du langage XML (ce dernier est très étendu et nous ne verrons dans ce document que les bases)

Dans un second temps, ce document présente aussi comment utiliser également les éléments suivants qui font partie du monde XML aussi:

1. XSL et XSD en utilisant le logiciel XMLSpy
2. Les flux RSS et leur traitement et création avec PHP et ASP 3.0/ASP.Net
3. Le XML dans les formulaires Adobe Acrobat 6.0
4. Le XML dans les animations Macromedia Flash
5. Le langage XSL-Flo pour la création de documents
6. Le langage Smil pour le multimédia
7. Les commandes de XPath et XLink
8. SVG
9. Enfin, des curiosités comme MathML (maths), CML (chimie, AML (astronomie), ThML (théologie), XBRL (eXtensible Business Reporting Language) pour la finance, KML (Google Earth), HL7 et HPRIM pour les échanges d'information dans le monde de la santé, etc.

Signalons qu'il existe une agence en France, appelé "ADeP" (Agence pourle Développement des e-Procédures), qui promeut des solutions pour les échanges d'informations entre administrations, entre les administrations et les citoyens, entre les collectivités locales et les notaires, etc. Le but est d'avance vers plus de dématérialisation, plus de productivité et plus de réactivité du service public. Au coeur de leur travail, on trouve la normalisation de schémas XSD que l'on étudiera plus tard en détails et par les outils MS Office InfoPath (voir mon livre électronique sur le sujet). Il y a aussi l'INSEE en france qui défini des schamas (voir l'adresse http://xml.inseee.fr/schema).

TABLE DES MATIÈRES

Préface

Avant toute chose, la présentation MS PowerPoint suivante (PPS) rédigée par votre serviteur pourra vous être utile:

et comme le contenu l'indique, il convient de rappeler avant toute chose que:

1. XML n'a pas été conçu prioritairement pour la création de pages web et de ce fait, ne prend en aucun cas la relève de HTML (du moins pas dans les prochaines années).

2. Contrairement au HTML, XML est en cours de développement. De nombreux éléments qui allaient de soi depuis des années avec HTML ne fonctionnaient tout simplement pas avec XML. Même les liens hypertextes ne fonctionnaient pas Internet Explorer 5.5.

3. XML seul ne représente aucune donnée, il ne fait que des les structurer.

4. Le XML pur n'est supporté que par les navigateurs les plus récents et voilà pourquoi il est inapproprié au web à ce jour.

Avoir affaire à une nouvelle technologie est quelque chose qui peut-être captivant. Une technologie qui ne s'imposera totalement que dans les prochaines années. Une technologie dont nous ne pouvons pas encore dire: "c'est ainsi", mais dont nous devons dire "ce sera ainsi", ou "cela pourrait donner ceci". Tout est encore en cours!

1. Liens Internet

Voici quelques liens Internet indispensables dans le domaine qui nous intéresse:

http://www.w3c.org
La référence mondiale (oblige!) dans le standard Web

http://www.w3schools.com
Petits cours en ligne sur toutes les possibilités de XML (bien plus que ce qui est présenté dans ce support) mais en anglais

http://xmlfr.org
Excellente ressource d'informations sur le XML francophone

http://www.codes-sources.com
Son nom indique son contenu (très impressionnant)

http://www.developpez.com
Référence francophone sur le développement avec les outils les plus courants du marché ! Vous y trouverez ce document.

http://www.phpscripts-fr.net

Tout sur le PHP "francophone". Certainement le meilleur site !

http://www.selfhtml.org/#fr
Vous saurez (presque) tout avec ce site sur l'HTML

http://msdn.microsoft.com
Pour tout ce qui est développement sur ou avec les produits Microsoft et contient bien sûr beaucoup de pages sur le XML

2. S.G.M.L.

La multiplicité engendre des problèmes ! Vous connaissez certainement ce drame dans le cadre d'échange de documents électroniques (autres que PDF) ou dans le cadre de la récupération d'archives électroniques (évolution des technologies oblige…).

Dans les faits, le monde de l'informatique est trop souvent comparable à la "tour de Babel". Chaque constructeur utilise son propre format pour ses programmes, format qu'il modifie régulièrement au bout de quelques années. Et même les filtres les plus perfectionnées ne parviennent pas toujours à transférer toute les propriétés d'un format vers un autre sans engendrer des pertes.

Si le système est aussi peu fiable que cela, que ferez-vous si, dans dix ans, vous voulez ouvrir les documents que vous avez créés aujourd'hui.

Ce problème n'est pas nouveau, au contraire, il profile déjà depuis des dizaines d'années. Voilà pourquoi dès les années 70, le Professeurs Charles F. Goldfab, sous les ordres d'IBM, s'est intéressé à la "description de documents". C'est ainsi que vit le jour le fameux *GML*, le *Generalized Markup Language*.

L'objectif du GML était de décrire des documents de manière à ce que le résultat ne dépende ni d'une plate-forme particulière, ni d'une application spécifique. Il ne s'agit pas de l'apparence d'un document mais de la structure du document: les titres, chapitres, pages, paragraphes, etc.

Au fil des ans, le GML a continué à être développé. C'est ainsi que naquit le *SGML*, *Standardized Generalized Markup Language*. Les développements allèrent si loin qu'en 1986, SGML pu être reconnu standard international, sous la désignation ISO 8879.

L'objectif de SMGL était de pouvoir enregistrer électroniquement et durablement des documents importants ou des systèmes d'exploitation en mutation constante.

SGML possédait un défaut, mais de taille: il était trop compliqué. Lors de son développement, on avait pensé aux administrations et aux services publics, mais pas au "texte normal" de l'utilisateur lambda. En conséquence, les outils logiciels de création SGML furent chers et très peu développés. Il est donc peu étonnant que, mises à par les institutions déjà évoquées, le développement de SGML fut très restreint.

Mais le World Wide Web naquit et le monde se transforma…

3. H.T.M.L.

Sans faire un cours sur l'HTML (qui est supposé connu pour le lecteur) il peut être intéressant de détailler les raisons pour lesquelles un nouveau langage de description tel que le XML était nécessaire.

Voyons donc les inconvénients de l'HTML:

1. HTML est conçu pour l'affichage dans un navigateur Web, donc moins adapté pour l'impression ou la mise en page. Les marges de la page ? Le format du papier ? La composition des colonnes ? Les lignes d'en-tête et de pied de page ? La table des matières, l'index et les renvois ? Malgré l'existence du langage mise en page CSS, il difficile de contrôler ces aspects avec le HTML.

2. Le stock de balises est limité, le langage n'est donc pas très flexible. Que faites vous quand vous avez, par exemple, besoin d'un graphique vectoriel ou de tableaux de cacluls dans vos projets? Le HTML ne le permet pas !

3. Le HTML seul ne peut pas afficher dynamiquement des contenus qui se modifient. La représentation de données dans des tableaux HTML est statique, et seuls des outils tels que des langages de script (Perl, PHP, etc.) peuvent vous aider à questionner le contenu d'une base de données et transférer vers des documents HTML qui sont modifiés en fonction des données à afficher.

4. Dans le HTML, la structure du document et les informations de mise en page sont parfois (peuvent être) mélangées.

Le dernier inconvénient cité est précisément l'inconvénient décisif ! Pourquoi ?

Simplement parce que HTML n'a pas été conçu pour réaliser une véritable mise en page, ses capacités de création sont très limitées. C'est pourquoi les concepteurs Web utilisent un langage de mise en page supplémentaire: le C.S.S. Mais fondamentalement, l'HTML est très limité pour "sortir" du web:

S vous prévoyez d'imprimer avec un peu de soin une page HTML (un catalogue par exemple), vous devez prévoir une page uniquement pour l'impression sur papier, en plus de la page web affichée à l'écran, idem pour l'affichage sur un écran de télé, ou d'un téléphone WAP, ou d'un PDA,…

C'est très fastidieux !

L'inverse est aussi vrai: les données issues de traitements de texte, de catalogues ou de bases de données doivent d'abord être transformées en HTML afin de pouvoir être affichées sur le web.

Ne serait-ce pas formidable de pouvoir disposer d'un format de base identique dans tous les cas ? Eh bien ce format est XML !

4. X.M.L.

Ce que SGML n'est pas arrivé à accomplir, sa bouture HTML, y est parvenu aisément: HTML est devenu un best-seller. Les entreprises, les institutions et un nombre toujours croissant de particuliers utilisent HTML pour placer des informations sur le web.

Les programmes d'affichage des pages HTML sont sans cesse améliorés, ainsi que les outils de création des pages.

Remarque: à vrai dire, les vrais professionnels préfèrent souvent coder les pages manuellement.

Et si HTML a désormais tellement de succès, pourquoi ne pas sauter de joie durant des heures et s'en tenir là ?

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

- une connaissance et une pratique aiguë du langage HTML.

- une connaissance et une pratique de la conception de pages Web.

- de bonnes notions de feuilles de style (CSS).

- des notions de Javascript ou de VBscript.

Le XML, en lui-même, ne fait rien !

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

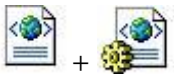
Voici un exemple de XML.

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

Ce qui affiché dans Internet Explorer donne le résultat suivant:

Pas que quoi fouetter un chat sur le plan esthétique... Le XML n'est que de l'information encodée entre des balises. Il faudra d'autres éléments, comme par exemple un fichier XSL,

pour que le navigateur puisse "comprendre" vos balises et afficher ce fichier sous une forme plus conviviale. D'où la partie de notre titre: XML plus XSL ou XML + XSL.



4.1 Limites et objectifs

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

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

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

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

Le XML serait plutôt un SGML qu'un HTML.

Le XML pour *eXtensible Markup Language* est donc un langage de balises (libre: ne nécessitant pas de licence de développement) comme le HTML mais il est extensible, évolutif. En XML, les balises ne sont pas prédéfinies. C'est vous qui devez ou pouvez définir vos propres balises. XML a été présenté en 1998 par le W3C, mais les premiers développements commencèrent dès 1996.

Et c'est là le problème! Si les braves navigateurs n'avaient plus de difficultés pour afficher les balises prédéfinies du HTML comme les <H1>, <BR> ou autres <TABLE>, que doivent-ils faire avec vos balises <ok> ou <new> ? Le XML a comme vocation de décrire de

l'information et pas d'afficher celle-ci. Ainsi le XML pourtant créé en 1999, est resté durant près de deux ans, un concept plutôt abstrait et théorique faute de moyens fiables pour en afficher le résultat. Avec le développement de nouvelles techniques comme le XSL, il est devenu possible de percevoir concrètement les énormes potentialités de ce nouveau langage.

Ainsi, nous pouvons résumer la situation par:

- HTML: nombre fini de balises
- XML: possibilité de définir ses propres balises

- HTML: balises pour formater
- XML: balises pour structure

Le XML a un petit inconvénient cependant: il est au format texte et nécessite dès lors une relative quantité de mémoire. Mais cet inconvénient est léger car la mémoire est de moins en moins chère. Et l'utilisation de la compression ZIP ou ARJ permet d'économiser la place pour le stockage des fichiers.

Remarque: les développeurs de StarOffice ont intégré dans le programme une routine Zip de ce cab et idem pour InfoPath avec une routine Cab (implicitement Zip).

4.2 Avenir du XML

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

Seul bémol à ces prédictions euphoriques, est la relative inertie des navigateurs grands publics à permettre, courant 2001, l'exploitation de ces nouvelles prescriptions. Comme exemple on peut citer le MathML qui est un langage issu du XML et qui a pour vocation d'afficher les formules mathématiques. Ce langage bien que parfaitement défini est encore peu exploitable par défaut dans les navigateurs les plus connus. Il faut pour le faire fonctionner télécharger un Plug-In appelé "MathPlayer" que l'on trouve facilement sur Internet.

Citons également le module SMIL (Synchronized Multimedia Integration Language) utilisé pour l'intégration multimédia (TV, vidéo).

4.3 XML et l'HTML

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

Le XML et le HTML sont deux langages distincts !

Le seul point commun entre le HTML et le XML est qu'ils sont issus tous deux de la même "mère" soit le SGML Standardized Generalised Markup Language qui est le langage de référence en milieu professionnel pour tout ce qui concerne la gestion électronique des

documents. Ils sont donc, tous deux, des langages de balises [Markup Language]. Ils ont également des caractéristiques communes héritées du SGML qui sont de transporter sur le Web des données en mode texte [plain text], compatibles avec n'importe quelle plateforme logicielle.

Le XML et le HTML sont tous deux issus du SGML avec lequel ils partagent des caractéristiques communes

Ils fonctionnent avec des balises.
Ils sont indépendants de la plateforme.
Ils sont en mode texte [plain text].

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

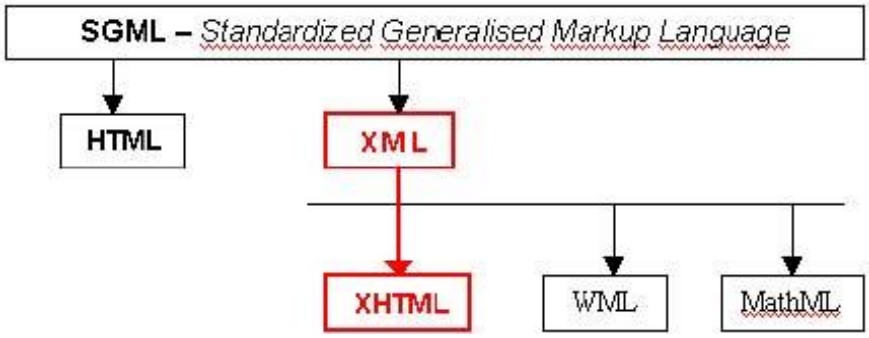
- Le XML décrit, structure, échange des données tandis que le HTML ne fait qu'afficher des données.

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

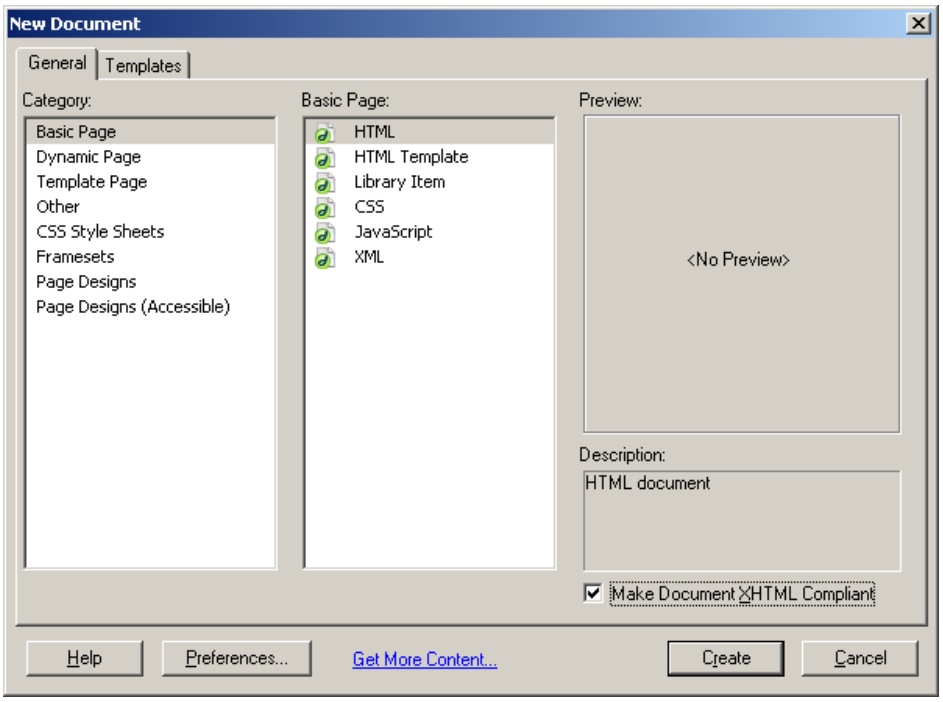
Le XML	Le HTML
- Le XML décrit structure, stocke, transporte et échange des données - Le XML est un générateur de langages [métalanguages] - Outre les PCs, le XML se veut adapté aux outils comme les mobiles, les pockets, etc. - Le XML est un langage strict dont l'écriture doit être rigoureuse	- Le HTML affiche des données par l'intermédiaire d'un navigateur - Le HTML est un langage statique (normalisé) de publication sur le web - Le HTML est surtout conçu pour les ordinateurs de type PC - Le HTML avec la version 4.0 est arrivé à bout de course et est devenu un langage hybride et en final peu structuré - Le HTML, à cause des navigateurs récents est devenu très primitif

4.4 HTML, XML et XHTML

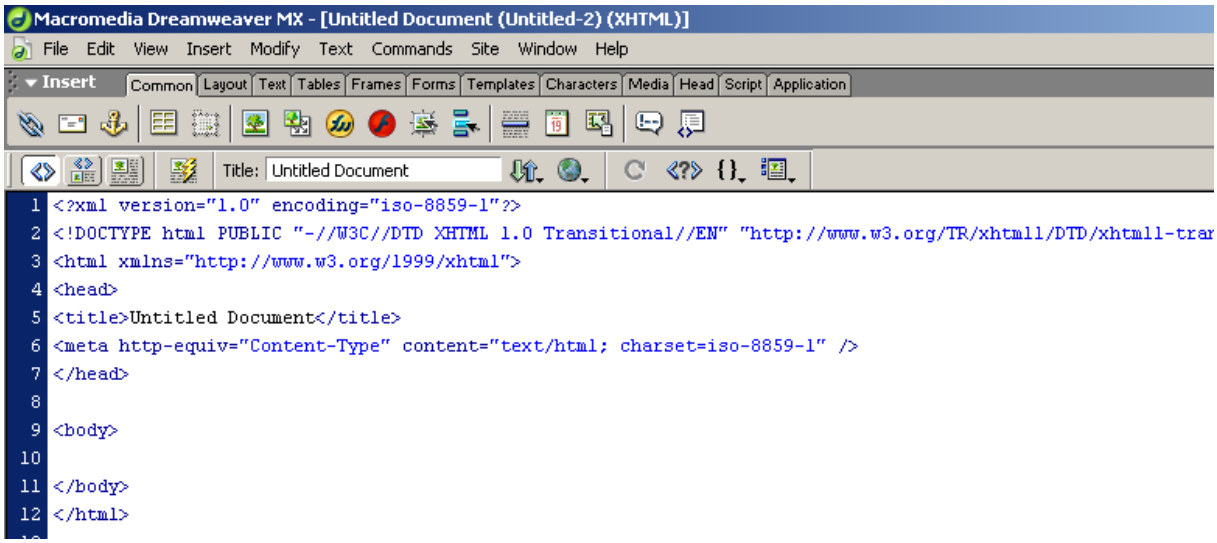
Le XHTML est quant à lui le successeur du HTML. Mais il est par ailleurs aussi un des "enfants" engendrés par le XML. En deux mots, pour faire un peu le ménage dans les dérives du HTML au fil des différentes versions, le W3C a conçu le XHTML qui n'est en fait qu'une reformulation du HTML 4.0 selon la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



Ceci est particulièrement visible lorsque l'on crée un nouvelle page XHTML dans Dreameaver:



On peut alors voir dans le code de la page:



On voit sur la première ligne la déclaration de la norme XML et sur la seconde, une référence

DTD.

4.5 Syntaxe XML

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

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

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

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

- Les noms peuvent contenir des lettres, des chiffres ou d'autres caractères.
- Les noms ne peuvent débuter par un nombre ou un signe de ponctuation.
- Les noms ne peuvent commencer par les lettres *xml* (ou XML ou Xml...).
- Les noms ne peuvent contenir des espaces.
- La longueur des noms est libre mais on conseille de rester raisonnable.
- On évitera certains signes qui pourraient selon les logiciels, prêter à confusion comme "-", ";", ".", "<", ">", etc.
- Les caractères spéciaux pour nous francophones comme é, à, ê, ï, ù sont à priori permis mais pourraient être mal interprétés par certains programmes.

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

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

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

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

Toute balise ouverte doit impérativement être fermée.

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

Ainsi en HTML, ce qui suit est affiché correctement:

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

Le XML est beaucoup plus strict. On devrait avoir:

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

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

Les balises doivent être correctement imbriquées.

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

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

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

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

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

Tout document XML doit comporter une racine!

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

Par exemple:

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

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

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

Par exemple:

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

Remarque: nous verrons tout de suite que les valeurs de ce que nous appelons les "attributs" doivent toujours être mises entre des guillemets.

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

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

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

La bonne écriture est:

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

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

Voici un premier document XML.

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

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

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

On en profite généralement pour notifier le "character set" qui indique à l'interpréteur XML [Parser] le jeu de caractères à utiliser. Le jeu de caractères "ISO-8859-1" correspond à Latin 1. Pour nous francophones, l'avantage d'accepter la plupart des lettres avec des accents.

Dans le tableau suivant, nous avons regroupé les principaux standards ISO:

STANDARD ISO	CODE PAYS
UTF-8 (Unicode)	Jeu de caractères universel, mondial
ISO-8859-1 (Latin-1)	Europe occidentale, Amérique latine
ISO-8859-2 (Latin-2)	Europe centrale et orientale
ISO-8859-3 (Latin-3)	Europe du sud-est
ISO-8859-4 (Latin-4)	Scandinavie, pays Baltes
ISO-8859-5	Cyrillique
ISO-8859-6	Arabe
ISO-8859-7	Grec
ISO-8859-8	Hébreu
ISO-8859-9	Turc
ISO-8859-10	Langues lapone, nordique, esquimaude
EUC-JP ou Shift_JIS	Japonais

Vous pourrez vous amuser si vous le voulez à changer les jeux de caractères pendant les exemples données plus tard en remplaçant par exemple ISO-8859-1 par ISO-8859-3 ou ISO-8859-5. Vous obtiendrez aussitôt un message d'erreur de la part d'Internet Explorer !

Remarque: l'omission de l'attribut de codage génère également un message d'erreur. Car le navigateur part ensuite de l'unicode (UTF-8) et exige un "masquage de l'accent".

Le remplacement des accents s'effectue par des entités: les entités sont des caractères de remplacement. Une entité commence par "&" et se termine toujours par un point-virgule ";". Nous utilisons les codes de caractères des tableaux ISO pour afficher les "caractères d'échappement". Ainsi:

" correspond au guillemet double, "

&#60; correspond à la relation d'ordre inférieur, <

Seuls les cinq caractères suivants, également appelées "entités nommées", sont autorisées:

Caractère	Entité
<	&lt;
>	&gt;
&	&amp;
"	&quot;
'	&apos;

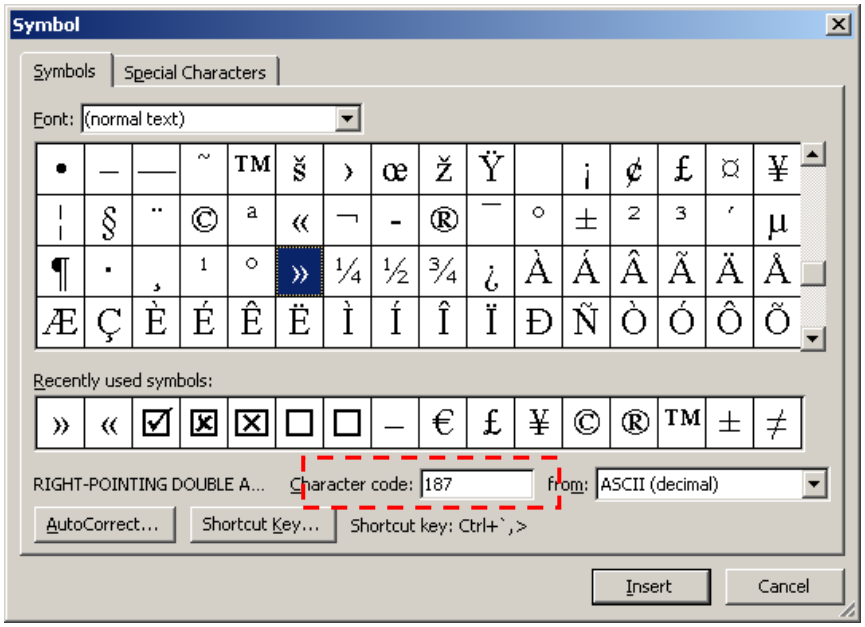
Les autres entités nommées telles que é ou è utilisées dans le HTML ne sont malheuresuement pas admises. Dans ce cas, utilisez les "codes".

Le tableau suivant présente quelques unes des caractères spéciaux les plus importants ainsi que l'entité correcpondante:

Caractère	Codage
»	&#187
«	&#171
©	&#169
®	&#174
£	&#163

Deux méthodes vous sont proposées dans ce document pour avoir la liste des autres codes.

1. Soit aller sur le site http://selfhtml.selfthtml.com.fr/th.htm
2. Dans MS Word aller dans le menu *Insertion* et choisir *Caractères spéciaux*:



Revenons-en à notre élément racine qui suit donc directement la déclaration du la norme de caractères la plus part du temps:

<racine>

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

... suite du document XML ...

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

</racine>

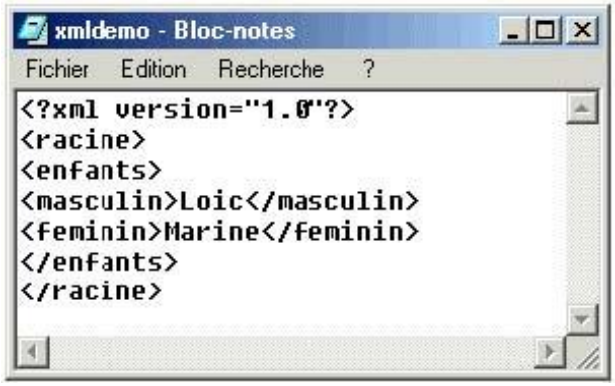
Le document XML se termine obligatoirement à la fermeture de la balise de racine.

Elaboration du fichier Voici un petit fichier XML.

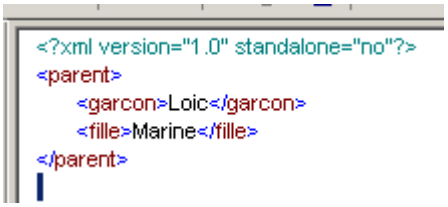
```
<?xml version="1.0"?>
<racine>
```

```
<enfants>
<masculin>Loic</masculin>
<feminin>Marine</feminin>
</enfants>
</racine>
```

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



Ou dans XMLSpy pour les utilisateurs de ce logiciel (qui a l'avantage, parmid d'autres, de fermer automatiquement les balises ouvertes …):



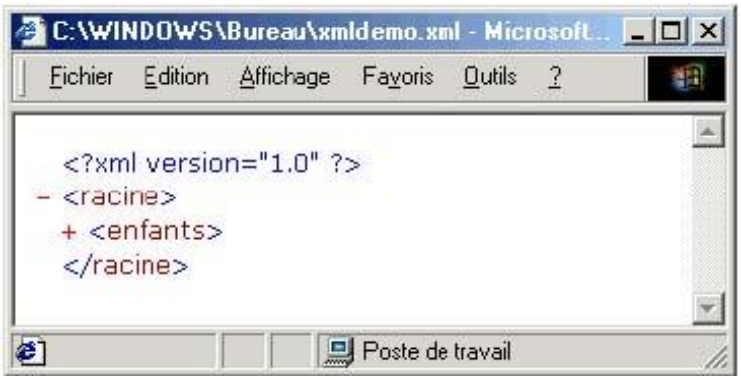
Et on l'enregistre (non pas en type de document Texte) en " Type: Tous (*.*) " sous un nom avec une extension .xml

Résultat dans Microsoft Explorer 5 et +

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

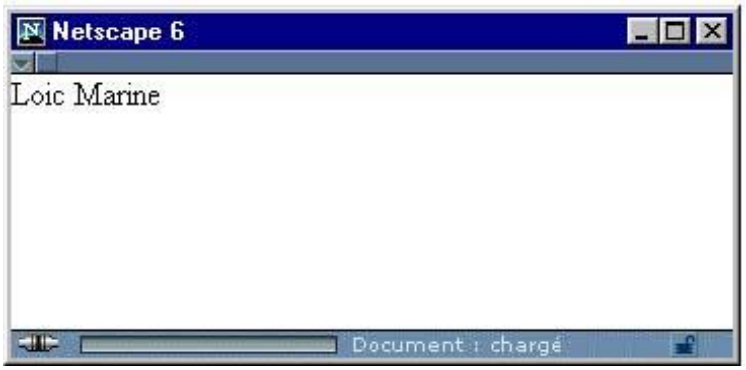
Remarque: Internet Explorer utilise une feuille XSL par défaut si aucune n'est référencée dans le document XML. Netscape 6 également, mais en sortie, il n'affiche que le texte. Mozilla et Firefox on également une feuille de style par défaut.

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



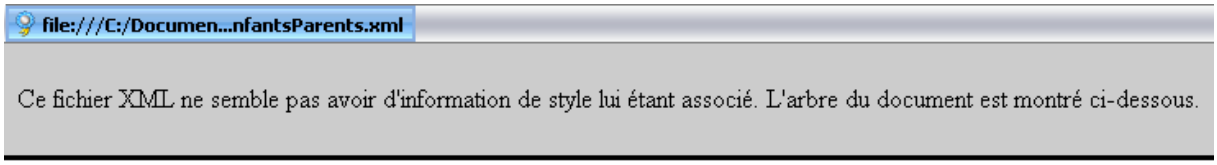
Résultat sous Netscape 6 et +

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



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

Regardons quand même du côté de Firefox 1.5 aussi:




```
- <racine>
  - <enfants>
      <masculin>Loic</masculin>
      <feminin>Marine</feminin>
    </enfants>
  </racine>
```

4.5.1 La balise CDATA (character data)

CDATA (pour "Character Data", données de caractère) figure dans les fichiers XML sous la forme ![CDATA[.

Pour résumer, le contenu de cette section ne sera pas analysé par le navigateur lors de la lecture du fichier. Le texte sera donc affiché tel qu'il est écrit dans cette section.

Nombre de fichiers XML dynamiques mettent leur contenu textuel dans une section CDATA, afin d'éviter qu'un caractère inattendu ne casse la validité du XML, et donc ne le rende illisible.

Cette section sert donc de plus ou plus en plus souvent de fourre-tout pour certains développeurs qui ne prennent pas soin de valider leur XML et préfère simplement cette section si pratique. Elle est donc utilisée parfois à tort ou à travers.

Exemple:

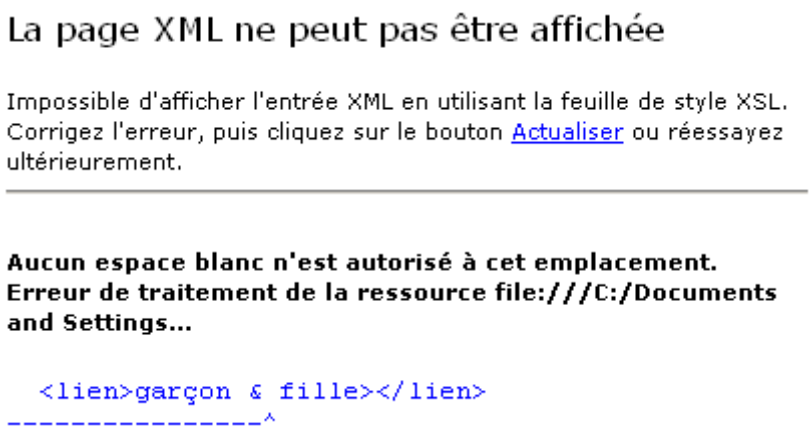
Créez un fichier nommé CData.xml suivant et affichez le dans le navigateur Internet Explorer:

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

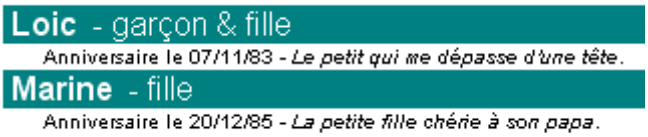
dans IE cela donne:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
- <racine>
  - <![CDATA[
                <enfant>
                        <nom>Loic</nom>
                        <lien>garçon & fille</lien>
                        <date>07/11/83</date>
                        <data>Le petit qui me dépasse d'une tête.</data>
                </enfant>

    ]]>
  - <enfant>
      <nom>Marine</nom>
      <lien>fille</lien>
      <date>20/12/85</date>
      <data>La petite fille chérie à son papa.</data>
    </enfant>
  </racine>
```

Associé à un fichier XSL que nous créerons plus loin en tant qu'exercice, le fichier XML sans la balise CDATA aura l'effet suivant à cause de la présence du caractère "&":



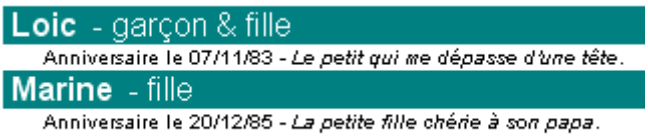
Si nous remplacons le "&" par son caractère d'échapement & nous obtiendrions:



En mettant le nœud cette fois-ci entre CDATA tel que ci-dessous:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml:stylesheet type="text/xsl" href="XStyleSheet.xsl"?>
<racine>
    <enfant>
        <nom>Loic</nom>
        <lien><![CDATA[garçon & fille]]></lien>
        <date>07/11/83</date>
        <data>Le petit qui me dépasse d'une tête.</data>
    </enfant>
    <enfant>
        <nom>Marine</nom>
        <lien>fille</lien>
        <date>20/12/85</date>
        <data>La petite fille chérie à son papa.</data>
    </enfant>
</racine>
```

Nous obtiendrons à nouveau:



La balise CData sert donc à ignorer tous les caractères spéciaux et permet souvent aux développeurs d'intégrer des scripts (du javascript la plupart du temps) dans des fichiers XML. Effectivement, les langages de scripts dynamiques contiennent très souvent trop de caractères spéciaux pour que l'on ait l'envie de tous les rempalcer par leur caractère d'échappement.

4.6 Le DTD et le XSD

Avant de vous lancer dans un projet XML, vous devez toujours
commencer par planifier la stucutre dans le but de la normaliser pour des usages futurs !

Définition: Le DTD (Document Type Declaration) ou encore XSD (eXtended Schema Document) est l'ensemble des règles et des propriétés que doit suivre le document XML.

Ces règles définissent généralement le nom et le contenu de chaque balise et le contexte dans lequel elles doivent exister. Cette formalisation des éléments est particulièrement utile lorsqu'on utilise de façon récurrente des balises dans un document XML. L'utilisation des fichiers XSD (language défini en 2001 par le W3C) est obligatoire pour un usage rigoureux de l'XML dans MS Office 2003 (le DTD étant plutôt opté par les anciens spécialistes Java – les vieux de la vieille quoi…)

Remarques:

R1. XSD est une évolution naturelle des DTD. Contrairement à DTD c'est un langage XML, il permet une description formelle des types, des namespaces (espaces de noms), des contraintes sur les données, etc... il est de ce fait plus adapté au développement d'applications (en java ou autre par les développeurs au courant…)

R2. Le fait que le XSD remplacera le DTD est du au fait que nous ne pouvons pas avec de dernier déterminer avec précision le type de données auquel nos caractères doivent

correspondre. Une restriction appliqée aux chiffres, au format monétaire ou même aux nombres entiers (integer) est impossible à ce jour (tout est pris dans #CDATA avec le DTD).

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

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

Voici une ficher XML faisant appel à son validateur XSD et son validateur XSD lui-même…:

Le fichier XML qui contient sa référence à un XSD de validation:

```
<?xml version="1.0" standalone="no"?>
<parent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="parent.xsd">
    <garcon>Loic</garcon>
    <fille>Marine</fille>
</parent>
```

Remarque: dans XMLSpy, lorsque l'on crée un nouveau fichier XML il vous demande automatiquement d'y associer un fichier XSD. Cela vous évitera d'écrire la pénible déclaration du namespace xsi

Le fichier XSD correspondant (nous verrons lors de notre étude de MS Office 2003 comment créer ce type de fichier très facilement):

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by isoz (sciences) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="parent">
        <xs:annotation>
            <xs:documentation>Comment describing your root element</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="garcon" type="xs:string"/>
                <xs:element name="fille" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Le fichier XML (à nouveau) qui contient sa référence à une DTD de validation:

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

et son DTD correspondant:

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

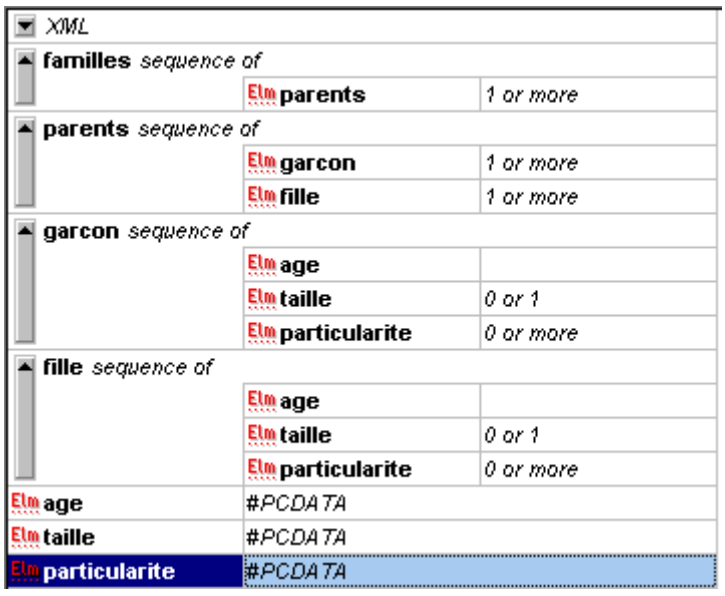
Nous pouvons trouver dans les DTD les symboles suivants:

	L'élément doit être présent au minimum une fois	A+
*	L'élément peut être présent plusieurs fois (ou aucune)	A*
?	L'élément peut être optionnellement présent	A?
|	L'élément A ou l'élément B peuvent être présents	A|B
,	L'élément A doit être présent et suivi de l'élément B	A,B
()	Les parenthèses permettent de regrouper des éléments afin de leur appliquer les autres opérateurs	(A,B)+

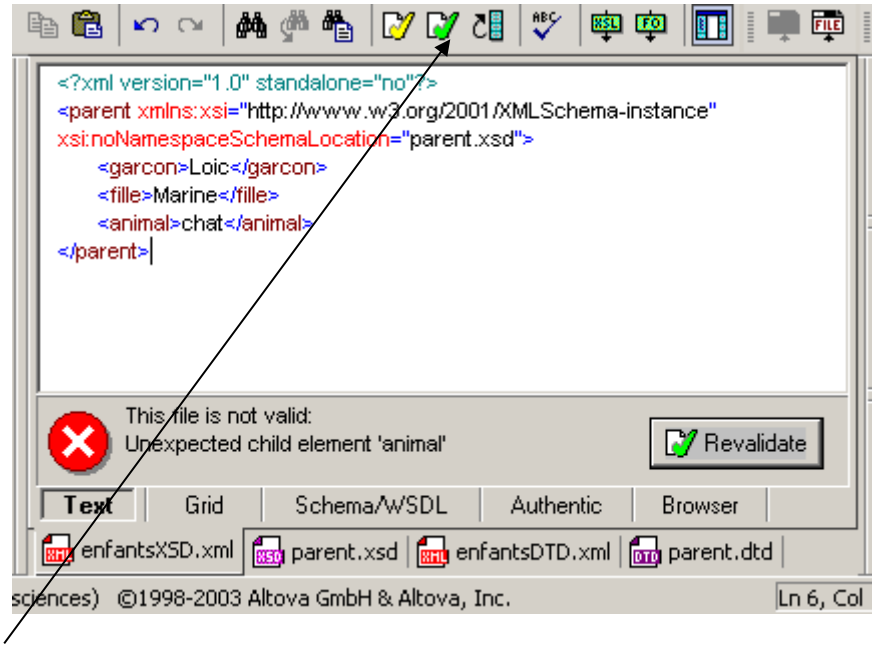
Exemple plus complexe:

```
<!ELEMENT familles (parents+)>
<!-- Cela va bien marcher -->
<!ELEMENT parents (garcon+, fille+)>
<!ELEMENT garcon (age, taille?, particularite*)>
<!ELEMENT fille (age, taille?, particularite*)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT taille (#PCDATA)>
<!ELEMENT particularite (#PCDATA)>
<!ATTLIST age
    annee CDATA  #REQUIRED
    lieu CDATA  #IMPLIED
>|
```

Ainsi, le nœud racine est familles et il peut contenir plusieurs nœuds parents qui lui-même peut contenir plusieurs nœuds garcon et taille… etc. Ce qui donne dans XMLSpy:

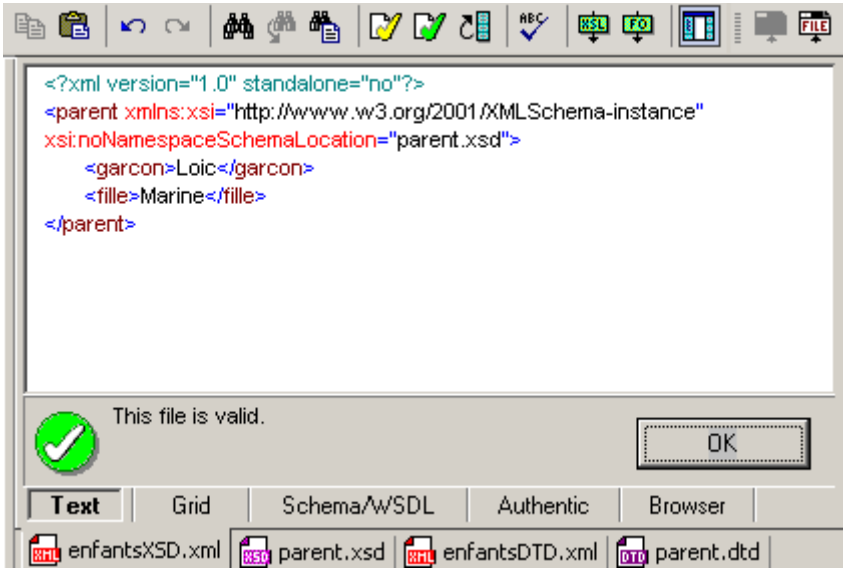


Revenons à l'exemple XML utilisant le XSD. Si nous le changeons de la manière suivante:



Dans XMLSpy un validateur vérifie (le jaune vérifie la conformité, le vert la validité) si le fichier XML est valide. En l'occurrence ce n'est plus le cas puisque la balise <animal> n'existe pas dans le XSD.

Si nous revenons à l'état initial du fichier XML et tentons à nouveau une validation, nous verrons que celle est OK:



4.7 XML et CSS

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

Un exemple de XML + CSS:

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

Voici notre document XML de départ:

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

Affiché dans le navigateur, cela nous donne:

Tristounet !

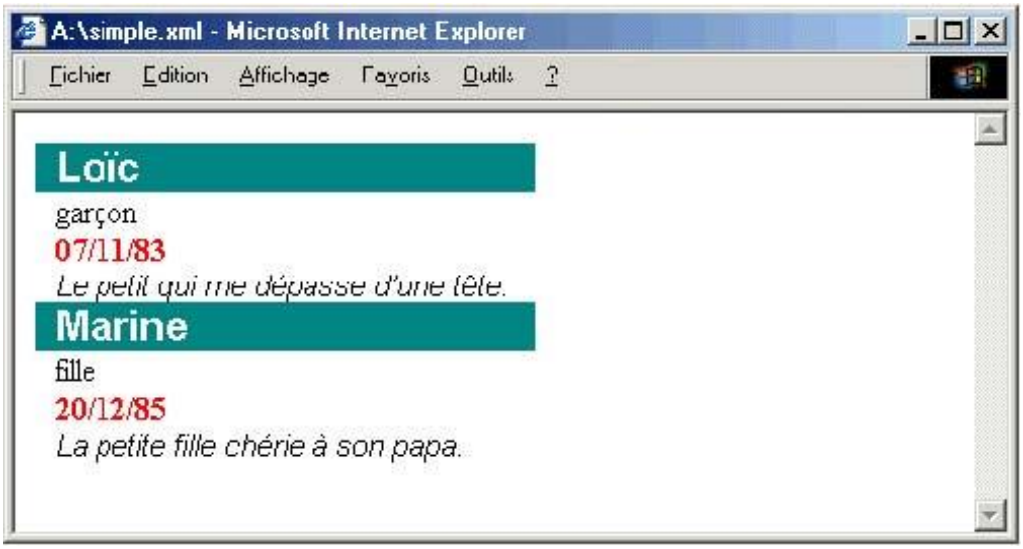
On crée le fichier .css dont voici le contenu:

```
/* CSS Document */
<style type="text/css">
racine.enfant{}
nom{
    display: block; width: 250px;
    font-size: 16pt; font-family: arial;
    font-weight: bold; background-color: teal;
    color: white; padding-left: 10px;
}
lien{
    display: block; font-size: 12pt; padding-left: 10px;
}
date{
    display: block; font-size: 12pt; color: red;
    font-weight: bold;  padding-left: 10px;
}
data{
    display: block; font-size: 11;
    font-style: italic; font-family: arial;
    padding-left: 10 px
}
</style>
```

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

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

On obtient finalement:



Plutôt sympa non ?...

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

4.8 XML et XSL

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

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

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

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

Voici notre document XML de départ:

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <?xml-stylesheet type= "text/xsl" href= "XStyleSheet.xsl"?>
3  <racine>
4      <enfant>
5          <nom>Loic</nom>
6          <lien>garçon</lien>
7          <date>07/11/83</date>
8          <data>Le petit qui me dépasse d'une tête.</data>
9      </enfant>
10     <enfant>
11         <nom>Marine</nom>
12         <lien>fille</lien>
13         <date>20/12/85</date>
14         <data>La petite fille chérie à son papa.</data>
15     </enfant>
16 </racine>
```

Affiché dans le navigateur, cela nous donne:

Pas très folichon !

On ajoute un fichier .xsl dont voici le contenu:

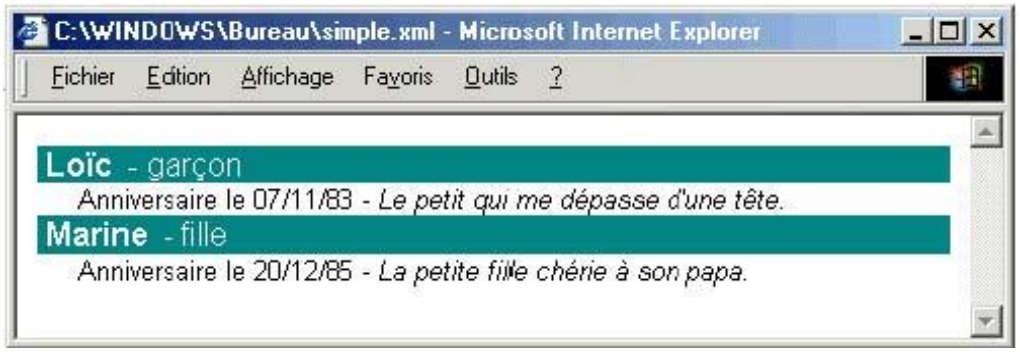
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body style="font-family:Arial; font-size:12pt;">
    <xsl:for-each select="racine/enfant">
    <div style="background-color:teal; color:white;">
    <span style="font-weight:bold; color:white; padding:4px">
    <xsl:value-of select="nom"/></span>
     - <xsl:value-of select="lien"/>
    </div>
    <div style="margin-left:20px; font-size:10px">
    <span>Anniversaire le <xsl:value-of select="date"/>
    </span>
    <span style="font-style:italic"> - <xsl:value-of select="data"/>
    </span>
    </div>
    </xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Attention!! Remarquez au débutl l'appel l'espace de noms (xmlns) de 1999 qui peut réserver quelque suprises au niveau de la syntaxe de certaines commandes XSL. Nous y reviendrons par ailleurs.

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

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

On obtient finalement:



Un peu mieux assurément !

4.9 XML dans HTML

On peut toujours incorporer du XML dans un fichier HTML avec la balise <xml> ... </xml>. Mais en toute logique, quand les navigateurs rencontrent des balises incorrectes ou inconnues, rien n'est affiché. Ce sera le cas avec vos balises XML incorporées dans un fichier HTML.

Heureusement, on peut passer par une astuce qui répond au doux nom romantique de "îlots de données" [Data Islands].

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

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

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

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

Vite, vite, un exemple !

Voilà toujours notre fichier XML (extérieur) qui ne doit pas faire référence à un fichier XSD!!:

Ne marche plus avec les navigateurs "modernes" depuis les années 2006 à 2015…

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

Soit:

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

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

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

Grandiose !

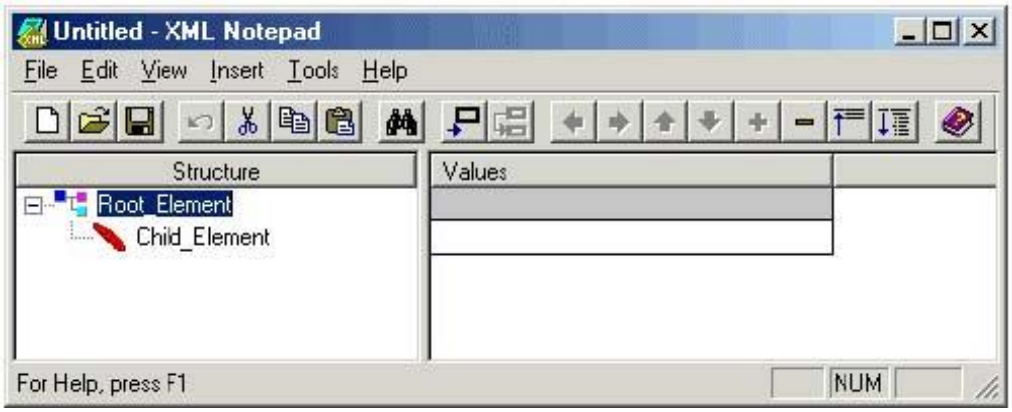
4.10 Editeurs XML

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

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

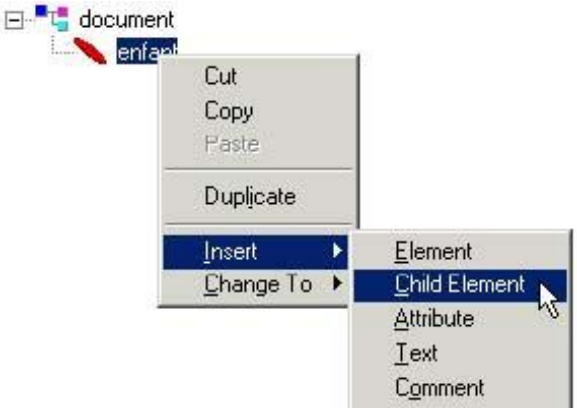
4.10.1 MXML Notepad

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



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

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



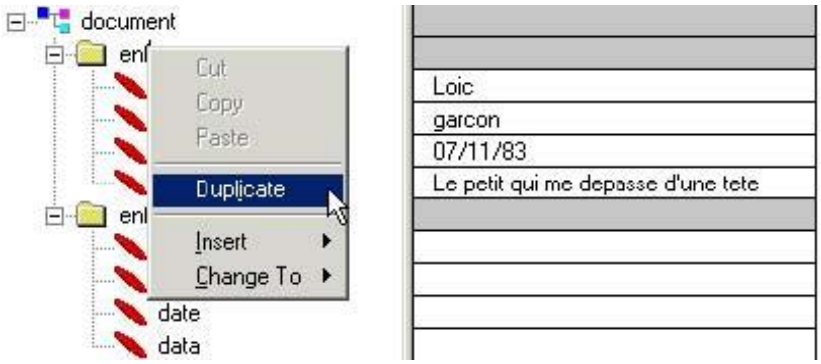
Dans la fenêtre Structure, le curseur d'insertion clignote dans le sous-élément ajouté. On y ajoute la balise *nom*.

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

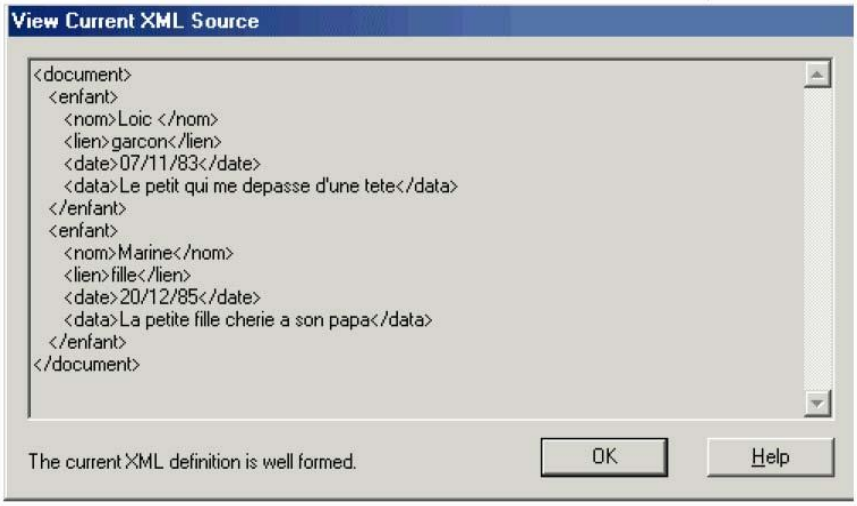
Ajoutons les autres sous-éléments *lien*, *date* et *data*. On clique sur Nom ♏♦ bouton droit et *Insert*/*Element* et on encode les données.

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

On peut encoder les valeurs.



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

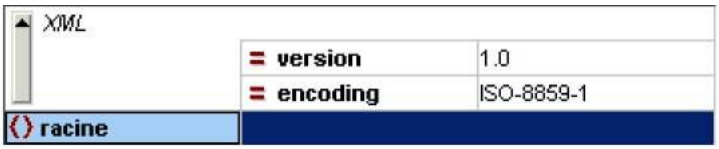


4.10.2 XMLSpy

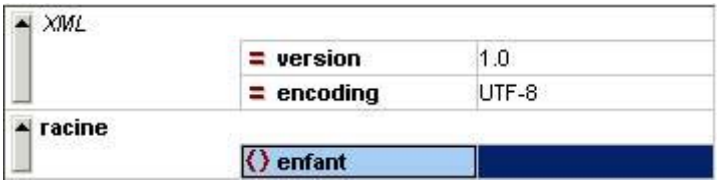
Avec XMLSpy (mon préféré), on dispose d'un programme déjà plus professionnel. Il est à notre avis assez remarquable pour la structure qu'il met en place.

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

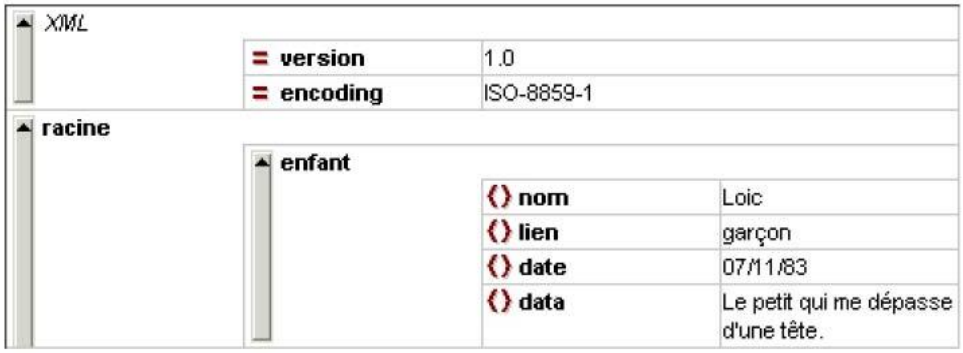
Commençons par l'élément racine. On prend le menu *XML/Insert/Element* et on encode la balise racine.



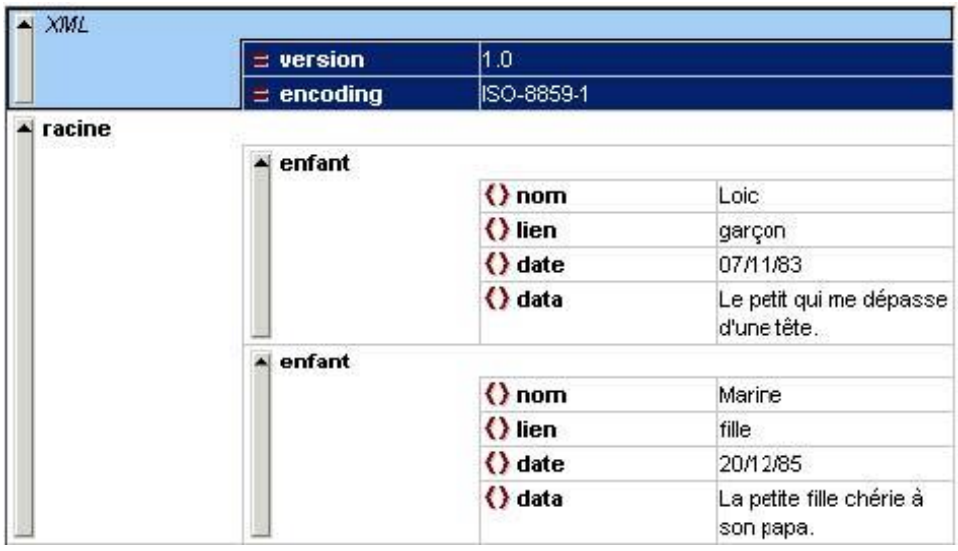
Encodons l'élément enfant, après avoir cliqué dans la zone racine *XML/Add child*



On passe aux sous-éléments. Après avoir cliqué dans la zone enfant, on demande 4 éléments soit menu: *XML / Add Child / Element* (4 fois).



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



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

Remarque: perso je préfère taper le XML à la main dans le mode texte de XMLSpy

4.10.3 Xmetal

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

5. X.S.L.

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

- une connaissance basique du XML abordé dans le chapitre précédent.
- une connaissance et une pratique aiguë du langage HTML.
- une connaissance et une pratique de la conception de pages Web.
- de bonnes notions de feuilles de style (CSS).
- des notions de Javascript ou de VBscript.

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

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

Remarque: une transformation XSLT transforme en réalité un document XML en un autre document XML. Si le document de sortie n'utilise que des balises HTML, on obtient un document XHTML.

Voici un exemple de XML:

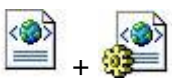
```
<?xml version="1.0"?>
<demoXML>
    <message>Voici du XML</message>
</demoXML>
```

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

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

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

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



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

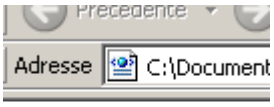
```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="fichierxsl.xsl"?>
<demoXML>
    <message>Voici du XML</message>
</demoXML>
```

Voici le fichier XSL:

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

Attention! Remarquez que nous n'utilisons plus dans cet exemple le vieux xmlns de 1999 mais le plus récent. Dans cet exemple cela n'a aucune importance mais plus tard…

Le résultat dans le navigateur est alors…:



Voici du XML

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

Le XSL ne fait pas que cela !

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

1. De sélectionner une partie des éléments XML (via des conditions ou des filtres ou XPath par la commande *select*)
2. De trier des éléments XML.
3. De filtrer des éléments XML en fonction de certains critères.
4. De choisir des éléments.
5. De retenir des éléments par des tests conditionnels.

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

Le XSL comporte en fait 3 langages:

1. Le XSL**T** qui est un langage qui **T**ransforme un document XML en un format, généralement en HTML, reconnu par un navigateur.

2. Le Xpath qui permet de définir et d'adresser des parties de document XML (nous traiterons ce langage dans les détails plus tard)

3. Le XML Formater (par l'espace de noms *fo*) pour "formater" du XML (transformé) de façon qu'il puisse être rendu sur des PocketsPC ou des unités de reconnaissance vocale.

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

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

Reprenons en bref:

- les balises sensibles à la casse, s'écrivent en minuscules.

- toutes les balises ouvertes doivent être impérativement fermées.

- les balises vides auront aussi un signe de fermeture soit <balise/>.

- les balises doivent être correctement imbriquées.

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

- le document XSL devra être "bien formé" [Well-formed].

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

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

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

- MSXML 2.0 est la référence du XML Parser de Internet Explorer 5.0 (mais pas compatible à 100%).

- MSXML 2.5 est la référence du XML Parser de IE 5.5 (en forte amélioration sans être parfait cependant).

- MSXML 3.0 est la dernière version du XML Parser. MSXML 3.0 peut déjà être téléchargé à partir du site de Microsoft et devrait être implémenté dans Internet

Explorer 6.0. A en croire les gens de Microsoft MSXML 3.0 et donc Internet Explorer 6.0 serait à 100% compatible avec les spécifications XML et XSL du W3C.

Cependant, avant de débuter, il est utile de préciser:

1. Que le XSL est dérivé du XML. Le document XSL reprend donc la structure et la syntaxe de n'importe quel document XML.

2. Que le document XSL comporte un document HTML ou XHTML qui sera quant à lui reconnu par le navigateur et qui servira de support à tout ou partie des données du document XML associé.

3. Que le XSL fonctionne avec une ou plusieurs "templates", sorte de gabarit pour définir comment afficher des éléments du fichier XML. Les éléments concernés du fichier XML sont déterminés par l'attribut "match".

Quelques informations préalables sur la syntaxe:

<?xml version="1.0"?>

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

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

La seconde ligne déclare que le document est du XSL extensible stylesheet. L'attribut xmlns fait référence au "namespace" utilisé.

Le namespace officiel du W3C est:

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

Pour la petite histoire 1999 fait référence à l'année d'apparition du concept XSL. Le xmlns (incorrect) de Microsoft IE soit:

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

est dû au fait que le XSL a été implanté dans Internet Explorer 5.0 alors qu'il n'était encore qu'en cours d'élaboration (WD pour working draft) par le W3C. Mais il a cependant l'avantage de contenir certaines fonctions utiles qui n'existent pas avec la 1999. Par ailleurs dans ce document nous jonglerons avec les deux afin de bien mettre en évidence les différences.

<xsl:template match="/">

Voila une balise template et son attribut match. Cette balise template va déterminer un gabarit dans lequel on va transformer des éléments du fichier XML sous une forme que le navigateur pourra afficher. Les éléments du fichier XML sont déterminés par l'attribut match="/". Le slash / entre guillemets signale que sont concernées toutes les balises XML du document (c'est du XPath) associé à partir de la racine [root].

<html> <body>

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

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

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

</body> </html>

Fin de la partie en HTML.

</xsl:template>

La fermeture de la balise de template.

</xsl:stylesheet>

Le document XSL se termine obligatoirement par la fermeture de la balise de déclaration de document XSL. Attention ! Pour que ce fichier XSL soit d'une quelconque utilité, il faut encore faire référence dans le fichier XML, au fichier XSL.

On ajoutera donc dans le fichier XML:

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

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

5.1 XSL: For-each et Order-By

Après cet aperçu théorique, étudions un exemple détaillé, soit une compilation de fichiers MP3. Voici un fichier XML que l'on reproduit dans le Bloc-notes ou Notepad (remarquez la référence au fichier OrderBy.xsl):

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="OrderBy.xsl"?>
<compilation>
    <mp3>
        <titre>Foule sentimentale</titre>
        <artiste>Alain Souchon</artiste>
    </mp3>
    <mp3>
        <titre>Solaar pleure</titre>
        <artiste>MC Solaar</artiste>
    </mp3>
    <mp3>
        <titre>Le baiser</titre>
        <artiste>Alain Souchon</artiste>
    </mp3>
    <mp3>
        <titre>Pourtant</titre>
        <artiste>Vanessa Paradis</artiste>
    </mp3>
    <mp3>
        <titre>Chambre avec vue</titre>
        <artiste>Henri Salvador</artiste>
    </mp3>
</compilation>
```

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

Passons maintenant au fichier XSL:

Le but de l'exercice est de représenter la compilation de mp3 sous forme d'un tableau trié dans l'ordre croissant des noms d'artistes (attention prenez garde au namespace utilisé ci-dessous sinon quoi le order-by ne fonctionnera pas, nous ne faisons donc pas usage de la norme de 1999!):

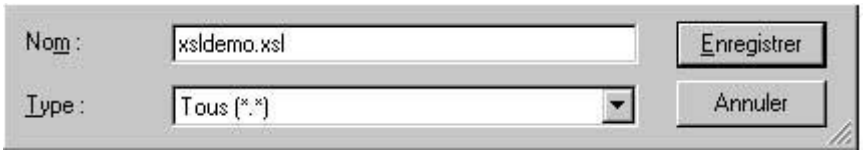
```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<html>
<body>
    <table border="1" cellspacing="0" cellpadding="3">
        <tr bgcolor="#FFFF00">
            <td>Artiste</td>
            <td>Titre</td>
        </tr>
        <xsl:for-each select="compilation/mp3"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

Attention! Vous remarquerez que nous n'utilisons encore une fois pas le xmlns de 1999. Essayez voir de le mettre et observez ce qu'il ce passe. Sinon il faut utiliser le **xsl:sort** avec oder="ascending" ou order="descending"!

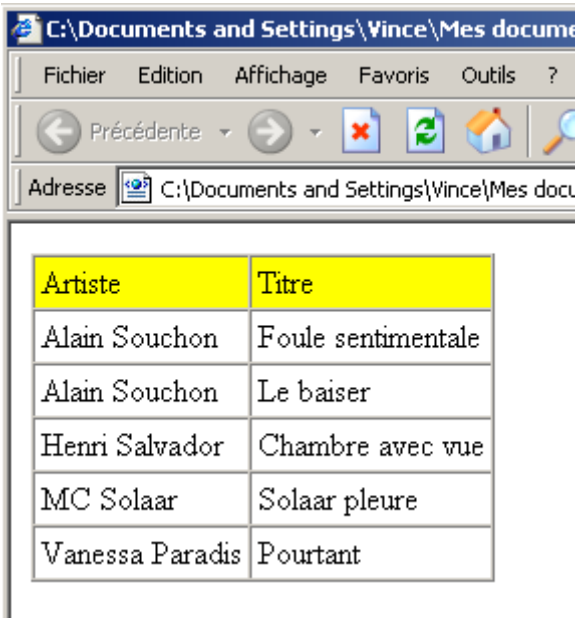
Remarque: L'avantage de xsl:sort par rapport à Order-By c'est que nous pouvons faire des tris à clés multiples.

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

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



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



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

5.2 XSL: Select

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

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

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

Pour ce faire, il suffira d'utiliser l'attribut select="chemin_d'accès[balise='xxx']". Attention à ne pas mettre d'apostrophes (' ') quand l'on fait un test sur une valeur numérique!!!

Les autres opérateurs possibles sont:

= pour égal , != pour différent (non égal), > pour plus grand que, < pour plus petit que

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

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

Elaboration du fichier

Prenons le fichier XML ci-dessous:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="ForEachSelect.xsl"?>
<compilation>
    <mp3>
        <titre>Foule sentimentale</titre>
        <artiste>Alain Souchon</artiste>
    </mp3>
    <mp3>
        <titre>Solaar pleure</titre>
        <artiste>MC Solaar</artiste>
    </mp3>
    <mp3>
        <titre>Le baiser</titre>
        <artiste>Alain Souchon</artiste>
    </mp3>
    <mp3>
        <titre>Pourtant</titre>
        <artiste>Vanessa Paradis</artiste>
    </mp3>
    <mp3>
        <titre>Chambre avec vue</titre>
        <artiste>Henri Salvador</artiste>
    </mp3>
</compilation>
```

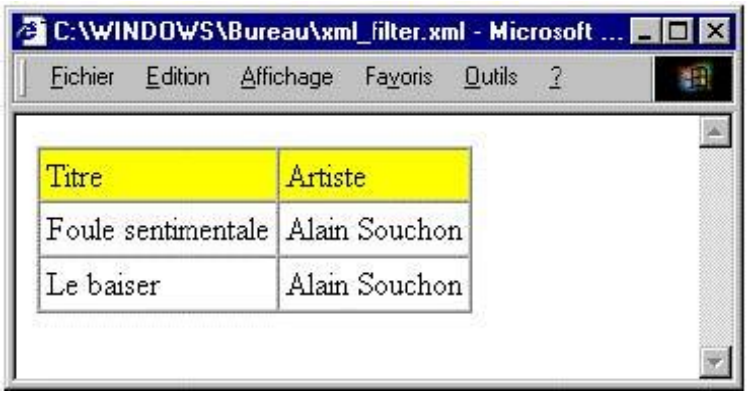
Passons maintenant au fichier XSL

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

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

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

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



Trop facile, non ?

5.3 XSL: If

Mais on peut aussi choisir avec le XSL en faisant usage des balises <xsl:if> ... </xsl:if> qui permettent d'effectuer un choix dans les données du fichier XML. On ajoutera l'attribut *match* où l'on indique l'élément choisi. Ce qui en résumé donne:

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

Il n'existe pas de xsl:else dans en XSL…

Elaboration du fichier Reprenons notre fichier XML (inchangé).

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="ifmatch.xsl"?>
<compilation>
    <mp3>
        <titre>Foule sentimentale</titre>
        <artiste>Alain Souchon</artiste>
    </mp3>
    <mp3>
        <titre>Solaar pleure</titre>
        <artiste>MC Solaar</artiste>
    </mp3>
    <mp3>
        <titre>Le baiser</titre>
        <artiste>Alain Souchon</artiste>
    </mp3>
    <mp3>
        <titre>Pourtant</titre>
        <artiste>Vanessa Paradis</artiste>
    </mp3>
    <mp3>
        <titre>Chambre avec vue</titre>
        <artiste>Henri Salvador</artiste>
    </mp3>
</compilation>
```

Passons maintenant au fichier XSL

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

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

On enregistre le fichier sous le nom ifmatch.xsl.

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

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

Attention!!!! Avec le standard 1999 du XSL la condition IF s'écrit:

<xsl:if test="balise='xxx'">balises HTM</xsl:if>

donc "match" est remplacé par "test" et les crochets ainsi que le point disparaissent.

5.4 XSL: Key

Il y a une autre manière de faire des conditions en utilisant des sortes de changement de variables à l'aide de l'instruction *key*. Voyons de quoi il s'agit en prenant toujours pour base notre fichier *simple.xml*:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="key.xsl"?>
<compilation>
    <mp3>
        <titre>Foule sentimentale</titre>
        <artiste>Alain Souchon</artiste>
    </mp3>
    <mp3>
        <titre>Solaar pleure</titre>
        <artiste>MC Solaar</artiste>
    </mp3>
    <mp3>
        <titre>Le baiser</titre>
        <artiste>Alain Souchon</artiste>
    </mp3>
    <mp3>
        <titre>Pourtant</titre>
        <artiste>Vanessa Paradis</artiste>
    </mp3>
    <mp3>
        <titre>Chambre avec vue</titre>
        <artiste>Henri Salvador</artiste>
    </mp3>
</compilation>
```

et maintenant, tout en prenant garde à changer la version des instructions XSL ainsi que la définition nous faisons la transformation consistant à prendre que les titre d'Alain Souchon mais en utilisant un changement de variable:

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

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!-- Attention ne pas oublier de definir la version 1.0 et le ns 1999 lors de l'utilisation de key -->

<xsl:key name="preg" match="mp3" use="artiste"/>

<xsl:template match="/">
<html>
<body>
    <table border="1" cellspacing="0" cellpadding="3">
        <tr bgcolor="#FFFF00">
            <td>Artiste</td>
            <td>Titre</td>
        </tr>
        <xsl:for-each select="key('preg','Alain Souchon')">
        <tr>
            <td><xsl:value-of select="artiste"/></td>
            <td><xsl:value-of select="titre"/></td>
        </tr>
        </xsl:for-each>
    </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Le résultat sera alors simplement:

Artiste	Titre
Alain Souchon	Foule sentimentale
Alain Souchon	Le baiser

L'instruction *key* est souvent utilisée dans le traitement de gros fichiers XML qui nécessitent souvent des instructions XPath longues. Ainsi, avec *key* l'instruction *XPath* est stockée dans l'attribut *use* et réutilisable à tout moment dans la transformation XSL.

5.5 XSL: Choose

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

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

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

```
                    </tr>
                    </xsl:otherwise>
        </xsl:choose>
```

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

Reprenons notre fichier XML (inchangé).

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="whenchoose.xsl"?>
<compilation>
    <mp3>
        <titre>Foule sentimentale</titre>
        <artiste>Alain Souchon</artiste>
    </mp3>
    <mp3>
        <titre>Solaar pleure</titre>
        <artiste>MC Solaar</artiste>
    </mp3>
    <mp3>
        <titre>Le baiser</titre>
        <artiste>Alain Souchon</artiste>
    </mp3>
    <mp3>
        <titre>Pourtant</titre>
        <artiste>Vanessa Paradis</artiste>
    </mp3>
    <mp3>
        <titre>Chambre avec vue</titre>
        <artiste>Henri Salvador</artiste>
    </mp3>
</compilation>
```

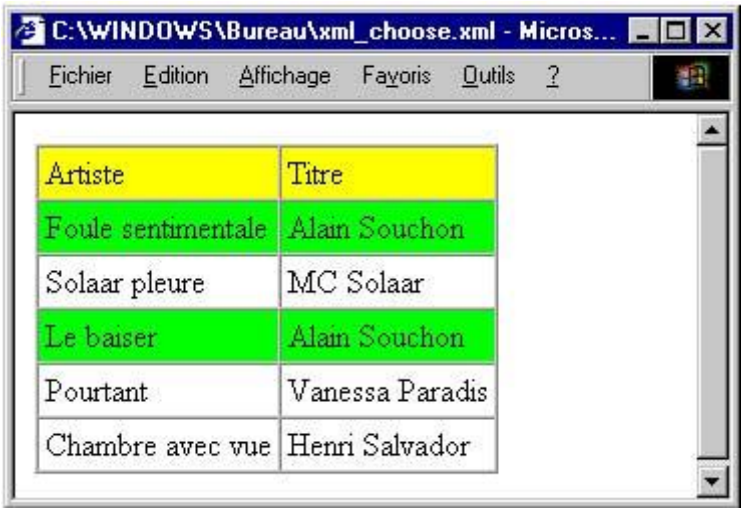
Passons maintenant au fichier XSL

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

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

On enregistre le fichier sous le nom whenchoose.xsl.

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



En tant qu'exercice, vous pouvez si vous le désirez rajouter dans le même fichier, un tri (by-order), un formatage conditionnel (when) et un filtre (if).

Attention!!! Idéalement il faudrait utiliser le nouveau standard tel que votre fichier XSL commence par:

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

et que la partie avec le test conditionnel s'écrive:

```
<xsl:when test="artiste='Alain Souchon'">
```

5.6 XSL: Variable

Nous allons voir ici une autre possibilité de XSL: la manipulation de variables.

Il est effectivement possible d'éffecteur quelques opérations élémentaires en XSL comme la division, la multiplication. l'addition, la soustraction, le calcul du modulo, etc.

A partir de ces éléments il est possible d'effectuer des choses remarquables proches des éléments élémentaires du JavaScript comme la création de calendriers, la génération de fractales, des calculs financiers et autres si le cœur, les possibilités et les connaissances nous le permettent.

Pour un exemple nous allons réutiliser un fichier XML déjà connu mais modifié un tantinet étant donné que maintenant il contient un nœud de prix.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="Variable.xsl"?>
<compilation>
    <mp3>
        <titre>Foule sentimentale</titre>
        <artiste>Alain Souchon</artiste>
        <prix>10.50</prix>
    </mp3>
    <mp3>
        <titre>Solaar pleure</titre>
        <artiste>MC Solaar</artiste>
        <prix>15.20</prix>
    </mp3>
    <mp3>
        <titre>Le baiser</titre>
        <artiste>Alain Souchon</artiste>
        <prix>13.50</prix>
    </mp3>
    <mp3>
        <titre>Pourtant</titre>
        <artiste>Vanessa Paradis</artiste>
        <prix>8.50</prix>
    </mp3>
    <mp3>
        <titre>Chambre avec vue</titre>
        <artiste>Henri Salvador</artiste>
        <prix>18.50</prix>
    </mp3>
</compilation>
```

A partir de ce fichier, nous souhaitons produire une table avec 3 trois colonnes pour chacun des informations présentes des les nœuds mp3 mais en y rajoutant une colonne calculée donnant la TVA et finalement avec une commande XPath, la somme totale.

Le fichier XSL correspondant est alors "simplement" le suivant:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
<html>
<body>
    <table border="1" cellspacing="0" cellpadding="3">
        <tr bgcolor="#FFFF00">
            <td>Artiste</td>
            <td>Titre</td>
            <td>Price</td>
            <td>TVA</td>
        </tr>
        <xsl:for-each select="compilation/mp3">
        <tr>
            <td><xsl:value-of select="artiste"/></td>
            <td><xsl:value-of select="titre"/></td>
            <td><xsl:value-of select="prix"/></td>
            <xsl:variable name="price" select="prix"/>
            <xsl:variable name="priceTVA" select="$price*0.076"/>
            <td><xsl:value-of  select="format-number($priceTVA,'0.00')"/></td>
        </tr>
        </xsl:for-each>
    </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Le résultat sera alors:

Artiste	Titre	Price	TVA
Alain Souchon	Foule sentimentale	10.50	0.08
MC Solaar	Solaar pleure	15.20	1.16
Alain Souchon	Le baiser	13.50	1.03
Vanessa Paradis	Pourtant	8.50	0.065
Henri Salvador	Chambre avec vue	18.50	1.41

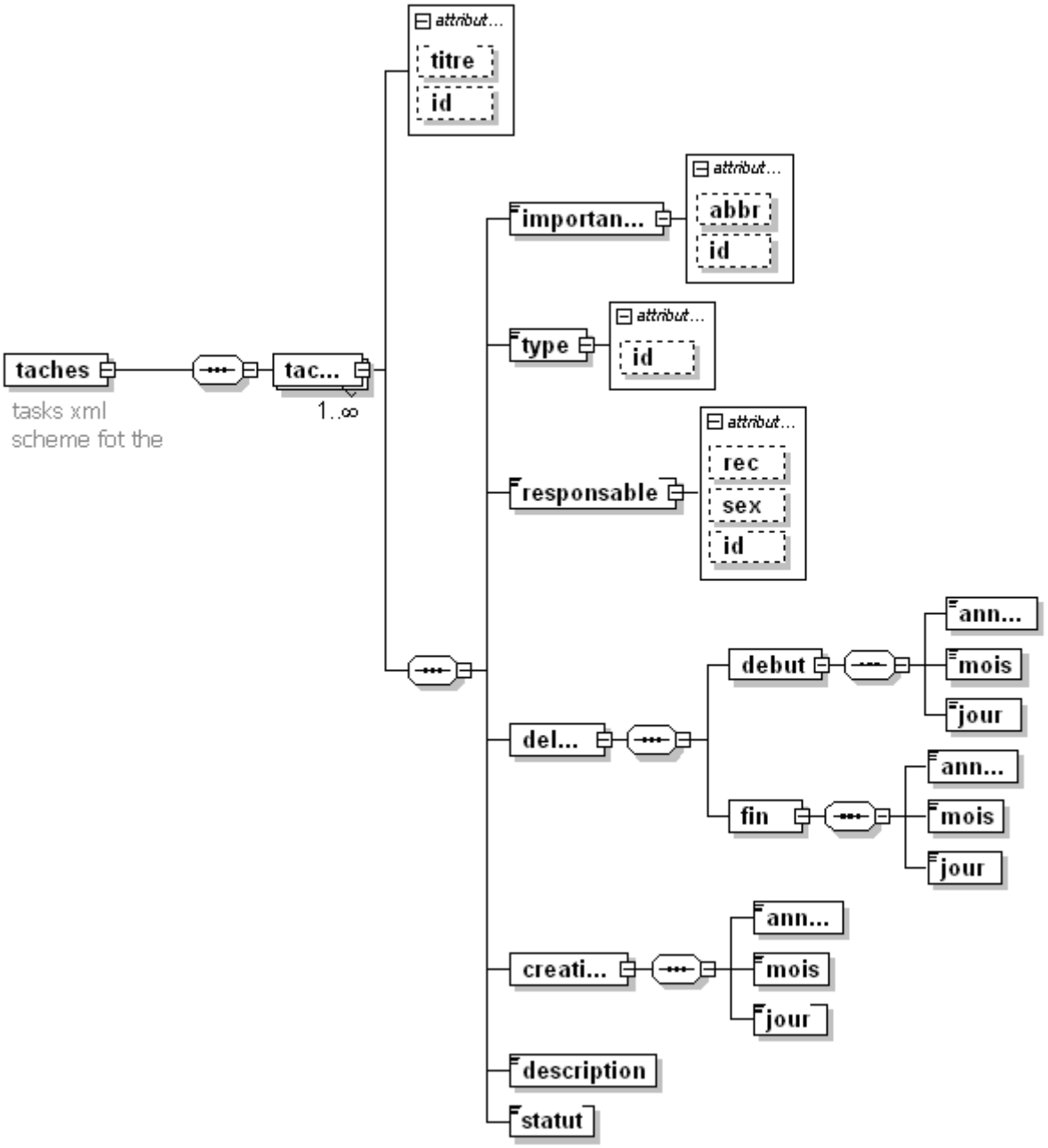
5.7 XSL: Attributs et images

Faisons un peu plus compliqué maintenant en utilisant également les "attributs" des balises, les tris, et les images. Soit le fichier XML suivant:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="affichage.xsl"?>
<taches>
    <tache titre="faire xml" id="204">
        <importance abbr="H" id="2">Haute</importance>
        <type id="3">Rangement</type>
        <responsable rec="manager" sex="female" id="3">Curie</responsable>
        <delais>
            <debut>
                <annee>2004</annee>
                <mois>02</mois>
                <jour>23</jour>
            </debut>
            <fin>
                <annee>2004</annee>
                <mois>02</mois>
```

```
                <jour>23</jour>
            </fin>
        </delais>
        <creation>
            <annee>2004</annee>
            <mois>02</mois>
            <jour>23</jour>
        </creation>
        <description>Apprendre XML, XSD et XSL</description>
        <statut>50%</statut>
    </tache>
    <tache titre="faire XSL" id="205">
        <importance abbr="H" id="2">Basse</importance>
        <type id="3">Rangement</type>
        <responsable rec="manager" sex="female" id="3">Isoz</responsable>
        <delais>
            <debut>
                <annee>2004</annee>
                <mois>02</mois>
                <jour>23</jour>
            </debut>
            <fin>
                <annee>2004</annee>
                <mois>02</mois>
                <jour>23</jour>
            </fin>
        </delais>
        <creation>
            <annee>2004</annee>
            <mois>02</mois>
            <jour>23</jour>
        </creation>
        <description>ha ha ha</description>
        <statut>50%</statut>
    </tache>
```

Dont le fichier XSD schématique correspondant est:



et le code XSD associé:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2006 sp2 U (http://www.altova.com) by isoz -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="taches">
        <xs:annotation>
            <xs:documentation>tasks xml scheme fot the company</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="tache" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="importance">
                                <xs:complexType>
                                    <xs:simpleContent>
                                        <xs:extension base="xs:string">
```

```
                                    <xs:attribute name="abbr" type="xs:string"/>
                                    <xs:attribute name="id" type="xs:integer"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="type">
                        <xs:complexType>
                            <xs:simpleContent>
                                <xs:extension base="xs:string">
                                    <xs:attribute name="id" type="xs:integer"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="responsable">
                        <xs:complexType>
                            <xs:simpleContent>
                                <xs:extension base="xs:string">
                                    <xs:attribute name="rec" type="xs:string"/>
                                    <xs:attribute name="sex" type="xs:string"/>
                                    <xs:attribute name="id" type="xs:integer"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="delais">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="debut"/>
                                <xs:element name="fin"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="titre" type="xs:string"/>
                <xs:attribute name="id" type="xs:integer"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Et traitons-le avec le fichier XSL donné ci-dessous et utilisant la syntaxe du xmlns de 1999 (on peut faire beaucoup mieux et cela pourrait être demandé par votre formateur en tant qu'exercice…):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
 <body>
   <h2>Mes taches</h2>
   <br/>
   <!--parcoure tous les champs et n'affiche qu'un manager donné-->
   <xsl:for-each select="taches/tache">
   <!--<xsl:for-each select="taches/tache[responsable='Curie']">-->
   <!--tri par importance-->
   <xsl:sort select="importance"/>
    <img src="pin.gif" align="absmiddle"></img>
    <i>Responsable:   </i><xsl:value-of select="responsable"/>
    <br/><br/>
    <i>Sexe:   </i><xsl:value-of select="responsable/@sex"/>
    <br/><br/>
    <i>Description:   </i><xsl:value-of select="description"/>
    <br/><br/>
    <xsl:if test="importance='Haute'">
```

```
        <font color="#FF0000">
        <img src="{importance}.gif" align="absmiddle"></img><xsl:value-of select="importance"/>
        </font>
    </xsl:if>
    <xsl:if test="importance='Basse'">
        <font color="#ff00ff">
        <img src="{importance}.gif" align="absmiddle"></img><xsl:value-of select="importance"/>
        </font>
    </xsl:if>
    <br/><br/>
   </xsl:for-each>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Le même fichier (pour la culture générale et c'est intéressant!) mais avec la syntaxe du namespace WD:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
  <html>
  <body>
    <h2>Mes taches</h2>
    <br/>
    <xsl:for-each select="taches/tache" order-by="+importance">
      <img src="pin.gif" align="absmiddle"></img>
      <i>Responsable:   </i><xsl:value-of select="responsable"/>
      <br/><br/>
      <i>Sexe:   </i><xsl:value-of select="responsable/@sex"/>
      <br/><br/>
      <i>Description:   </i><xsl:value-of select="description"/>
      <br/><br/>
      <xsl:if match=".[importance='Haute']">
          <font color="#FF0000">
            <img><xsl:attribute name="src"> <xsl:value-of select="importance/@img"/></xsl:attribute></img>
           <xsl:value-of select="importance"/>
          </font>
      </xsl:if>
      <xsl:if match=".[importance='Basse']">
           <font color="#FF0000">
          <img><xsl: attribute name="src"> <xsl:value-of select="importance/@img"/></xsl:attribute></img>
          <xsl:value-of select="importance"/>
          </font>
      </xsl:if>
      <br/><br/>
    </xsl:for-each>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

On voit que la syntaxe change nettement et en particulier tout ce qui concerne les attributs ! Par exemple (particuliers):

1999	WD
<img src="*NomDuNoeud* ">	<img><xsl: attribute name="src"> <xsl:value-of select="*NomDuNoeud*/*@NomAttributDuNoeud*"/> </xsl:attribute></img>
	<a><xsl: attribute name="name"> <xsl:value-of select=" *NomDuNoeud* "/> </xsl:attribute>
<option value="*@NomAttribut*">	<option><xsl: attribute name="value"> <xsl:value-of select="*@NomAttribut*"/> </xsl:attribute></option>

Dans le même dossier que celui où vous avez enregistré les fichiers tache.xml et affichage.xsl rajoutez les images suivantes (prenez garde au nom des images qui correspondent aux valeurs des balises <importance> de notre fichier XML d'origine):

L'affichage du traitement donne le résultat ci-dessous:

Pas mal non… et vous allez-voir on va faire encore mille fois mieux.

5.8 XSL: Grouping

Depuis XPath 2.0 il est possible de faire de manière assez simple une agrégation (regroupement) de données se trouvant dans un fichier XML (attention! il faut une version récente de XMLSpy pour tester l'exemple avec grouping… en tout cas ultérieure à 2006).

Considérons le fichier *products.xml* suivant:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="Transform.xsl"?>
<products>
    <product>
        <name>Crunch chocolate bar</name>
        <category>A</category>
        <price>10</price>
    </product>
    <product>
        <name>Cereals</name>
        <category>B</category>
        <price>20</price>
    </product>
    <product>
        <name>Lindt chocolate bar</name>
        <category>A</category>
        <price>20</price>
    </product>
</products>
```

Nous souhaiterions à partir de ce fichier XML générer un autre fichier XML dans lequel les produits sont rangés par catégories (pour générer un fichier XML à partir d'un fichier XML il n'est donc pas nécessaire d'avoir une version récente de XMLSpy par contre!).

Pour ce faire, il faudra créer le fichier *transform.xsl* suivant:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Attention ne marche qu'avec la version 2 du XSL -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes" version="1.0" encoding="UTF-8"/>

<xsl:template match="products">
<products>
   <xsl:for-each-group select="product" group-by="category">
      <category name="{current-grouping-key()}">

            <xsl:for-each select="current-group()">
               <product>
                  <xsl:value-of select="name"/>
               </product>
            </xsl:for-each>
      </category>
   </xsl:for-each-group>
   </products>
</xsl:template>

</xsl:stylesheet>
```

Remarquez la référence à la version 2.0 du namespace XSL et l'utilisation d'attributs de regroupement. Ensuite, dans XMLSpy, retournez dans le fichier XML et cliquez sur  pour que le parsage et la transformation en XML soit effectuée.

Vous obtiendrez alors le résultat suivant:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<products>
  <category name="A">
    <product>Crunch chocolate bar</product>
    <product>Lindt chocolate bar</product>
  </category>
  <category name="B">
    <product>Cereals</product>
  </category>
</products>
```

5.9 XSL: Templates, formulaires et javascript

Plus compliqué encore et toujours avec le même fichier XML et avec les mêmes petites images:

1. Nous allons créer un élément de formulaire avec une liste déroulante qui se remplit à partir du contenu du fichier XML et allant directement à la tâche choisie en utilisant du Javascript et des "ancres".

2. Dans le fichier XSL plutôt que d'afficher la liste des tâches simplement, nous allons créer un "modèle" de traitement à l'extérieur de l'affichage de la page web afin de pouvoir l'utiliser au besoin plusieurs fois en ayant à l'écrire qu'une seule fois !!! (comme les fonctions en programmation en quelque sorte)

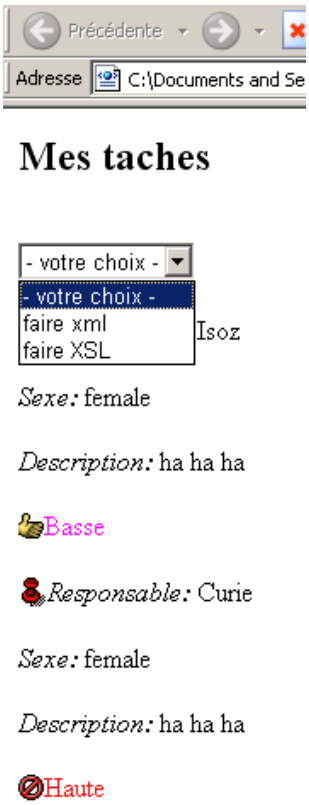
Voici le fichier XSL avec la norme de 1999[1] (nommez le fichiers selon votre convenance):

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!--Attention bien observer la ligne juste en-dessous pour plus tard!!-->
<xsl:template match="/">
 <html>
 <head>
 </head>

 <body>
   <h2>Mes taches</h2>
   <br/>
   <!--même chose que précédemment mais sans for each-->
   <form name="form1" method="post" action="">
     <select name="menu" onChange="document.location='#'+this.value">
       <option selected="">- votre choix -</option>
      <xsl:for-each select="taches/tache">
         <option value="{@id}"><xsl:value-of select="@titre"/></option>
      </xsl:for-each>
     </select>
   </form>
   <xsl:apply-templates select="taches/tache">
   <!--<xsl:apply-templates select="taches/tache[importance='Haute']">-->
       <xsl:sort select="importance"/>
   </xsl:apply-templates>
 </body>
 </html>
</xsl:template>


<xsl:template match="tache">
    <img src="pin.gif" align="absmiddle"></img>
```

[1] Attention! Si vous utiliser le WD il ne faut pas oublier de changer tous les attributs ah-hoc selon le tableau vu plus haut

```
    <a name="{@id}"></a>
    <i>Responsable:   </i><xsl:value-of select="responsable"/>
    <br/><br/>
    <i>Sexe:   </i><xsl:value-of select="responsable/@sex"/>
    <br/><br/>
    <i>Description:   </i><xsl:value-of select="description"/>
    <br/><br/>
    <xsl:if test="importance='Haute'">
        <font color="#FF0000">
        <img src="{importance}.gif" align="absmiddle"></img><xsl:value-of select="importance"/>
        </font>
    </xsl:if>
    <xsl:if test="importance='Basse'">
        <font color="#ff00ff">
        <img src="{importance}.gif" align="absmiddle"></img><xsl:value-of select="importance"/>
        </font>
    </xsl:if>
    <br/><br/>
</xsl:template>

</xsl:stylesheet>
```

Le résultat dans le navigateur étant:



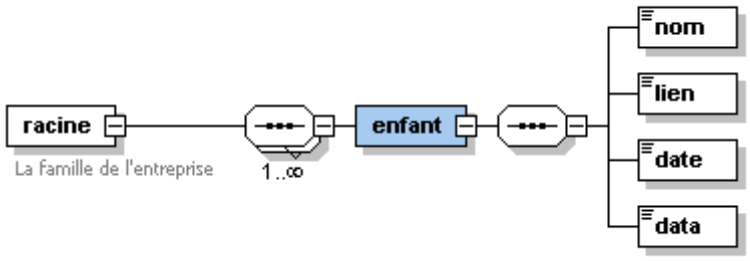
Nous voyons bien que les tâches de notre fichier XML sont toutes dans la liste et lorsque nous faisons un choix, le navigateur nous amène directement à celle-ci.

5.10 XSL: Substring et javascript

Nous allons maintenant revenir sur un exemple que nous avions fait au début de notre étude de l'XML mais de manière non rigoureuse. Soit le fichier XML suivant:

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

Maintenant, si vous faites un fichier XSD correspondant (voir le chapitre sur MS Word 2003 pour faire cela):




```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by Isoz Vincent (Sciences.ch) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="racine">
        <xs:annotation>
            <xs:documentation>La famille de l'entreprise</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
                <xs:element name="enfant">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="nom" type="xs:string"/>
                            <xs:element name="lien" type="xs:string"/>
                            <xs:element name="date" type="xs:date"/>
                            <xs:element name="data" type="xs:string"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Les dates étant au format 07/11/83 dans le fichier XML elles ne sont pas conformes donc le fichier XML ne pourra être validé !!!

Pour les rendre conforme, vous devez alors les changer tel que ci-dessous:

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="XSLStyleSheet.xsl"?>
<racine xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="Scheme.xsd">
    <enfant>
        <nom>Loic</nom>
        <lien>garçon</lien>
        <date>2004-10-25</date>
        <data>Le petit qui me dépasse d'une tête.</data>
    </enfant>
    <enfant>
        <nom>Marine</nom>
        <lien>fille</lien>
        <date>1985-12-20</date>
        <data>La petite fille chérie à son papa.</data>
    </enfant>
</racine>
```

La question est ensuite… mais comment dans le fichier XSL parser la date au format que nous voulons ??? Eh bien ce n'est pas compliqué et par ailleurs même important pour les gens travaillant avec MS InfoPath[2], voici le fichier XSL correspond pour afficher les dates au format jj.mm.aaaa:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <html>
            <body style="font-family:Arial; font-size:12pt;">
                <xsl:for-each select="racine/enfant">
                    <div style="background-color:teal; color:white;">
                        <span style="font-weight:bold; color:white; padding:4px">
                            <xsl:value-of select="nom"/>
                        </span>
    - <xsl:value-of select="lien"/>
                    </div>
                    <div style="margin-left:20px; font-size:10px">
                        <span>Anniversaire le
                            <xsl:value-of select="format-number(substring(date, 9, 2), '00')"/>
                            <xsl:text>. </xsl:text>
                            <xsl:value-of select="format-number(substring(date, 6, 2), '00')"/>
                            <xsl:text>. </xsl:text>
                            <xsl:value-of select="format-number(substring(date, 1, 4), '0000')"/>
                        </span>
                        <span style="font-style:italic"> - <xsl:value-of select="data"/>
                        </span>
                    </div>
                </xsl:for-each>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Attention!!! Le xmlns de 1999 contrairement au WD de Microsoft ne permet pas le parsage des nœuds avec préfixes (ou "espace de noms" abusivement). Dès lors, pour parser un fichier XML tel que celui-ci:

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml:stylesheet type="text/xsl" href="Parser.xsl"?>
<chantiers xmlns:cls="http://www.colas.ch">
    <cls:chantier>
```

[2] Logiciel de création de formulaires qui transforme automatiquement les dates au format américain dans le fichier XML résultant. Nous verrons par ailleurs un exemple plus loin de ce logiciel.

```
        <cls:responsable>Isoz</cls:responsable>
        <cls:mandataire>Confédération</cls:mandataire>
        <cls:duree>80</cls:duree>
        <cls:ressources>150</cls:ressources>
        <cls:lieu>Geneve</cls:lieu>
        <cls:date>2005-07-10</cls:date>
    </cls:chantier>
    <cls:chantier>
        <cls:responsable>Garcia</cls:responsable>
        <cls:mandataire>Commune</cls:mandataire>
        <cls:duree>50</cls:duree>
        <cls:ressources>150</cls:ressources>
        <cls:lieu>Geneve</cls:lieu>
        <cls:date>2006-05-31</cls:date>
    </cls:chantier>
    <cls:chantier>
        <cls:responsable>Miéville</cls:responsable>
        <cls:mandataire>Commune</cls:mandataire>
        <cls:duree>20</cls:duree>
        <cls:ressources>200</cls:ressources>
        <cls:lieu>Lausanne</cls:lieu>
        <cls:date>2006-01-12</cls:date>
    </cls:chantier>
</chantiers>
```

Le désir de déclarer le début du fichier XSL avec:

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

est plus que tentante (surtout pour les utilisateurs d'InfoPath qui intègre le préfixe *my* par défaut dans les formulaires). Le problème avec le WD c'est que la fonction substring que nous souhaiterions utiliser pour formater nos dates correctement n'est pas reconnue et renverra un message d'erreur.

Il faut alors ruser un peu et utiliser du Javascript tel que proposé ci-dessous pour l'exemple de nos chantiers:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<html>
<body>

<table width="50%">
<tr>
    <td>responsable</td>
    <td>mandataire</td>
    <td>duree</td>
    <td>lieu</td>
    <td>date</td>
</tr>
<xsl:for-each select="chantiers/cls:chantier">
    <tr>
        <td><xsl:value-of select="cls:responsable"/></td>
        <td><xsl:value-of select="cls:mandataire"/></td>
        <td><xsl:value-of select="cls:duree"/></td>
        <td><xsl:value-of select="cls:lieu"/></td>
        <td>
            <script>
                str="<xsl:value-of select="cls:date"/>".substring(5,7);
                document.write(str)
                document.write('.')
                str="<xsl:value-of select="cls:date"/>".substring(8,10);
                document.write(str)
                document.write('.')
                str="<xsl:value-of select="cls:date"/>".substring(0,4);
                document.write(str)
            </script>
```

```
        </td>
    </tr>
</xsl:for-each>
</table>

</body>
</html>
</xsl:template>

</xsl:stylesheet>
```

On remarquera la partie *script* qui utilise du Javascript pour formater les dates correctement!

5.11 Exercice: CSS ZenGarden

Et maintenant le plus fort (pour l'esthétique) mais en version light. Soit le fichier XML suivant (il ne contient que les titres des paragraphes de la page que vous allez voir mais nous laissons le lecteur faire de même pour tout le reste du texte – cela constitue un bon exercice par ailleurs):

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml:stylesheet type="text/xsl" href="Zen Garden.xsl"?>
<!-- les commentaires sont comme en html -->
<titres>
    <titre1>
        The Road to Enlightenment
    </titre1>
    <titre2>
        So What is This About?
    </titre2>
    <titre3>
        Participation
    </titre3>
    <titre4>
        Benefits
    </titre4>
    <titre5>
        Requirements
    </titre5>
    <titre6>
        Select a Design:
    </titre6>
    <titre7>
        Archives:
    </titre7>
    <titre8>
        Resources:
    </titre8>
</titres>
```

Et soit le fichier XSL suivant (alors là c'est vrai qu'il faut bien connaître l'XHTML…):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<HTML>
<HEAD>
<TITLE>css Zen Garden: The Beauty in CSS Design</TITLE>

<STYLE title="currentStyle" type="text/css">@import url(002.css );</STYLE>

</HEAD>
<BODY id="css-zen-garden">
<xsl:for-each select="titres">
<DIV id="container">
<DIV id="intro">
<DIV id="pageHeader">
```

```
<H1><SPAN>css Zen Garden</SPAN></H1>
<H2><SPAN>The Beauty of <ACRONYM title="Cascading Style Sheets">CSS</ACRONYM>
Design</SPAN></H2></DIV>
<DIV id="quickSummary">
<P class="p1"><SPAN>A demonstration of what can be accomplished visually through
<ACRONYM title="Cascading Style Sheets">CSS</ACRONYM>-based design. Select any
style sheet from the list to load it into this page.</SPAN></P>
<P class="p2"><SPAN>Download the sample <A
title="This page's source HTML code, not to be modified."
href="http://www.csszengarden.com/zengarden-sample.html">html file</A> and <A
title="This page's sample CSS, the file you may modify."
href="http://www.csszengarden.com/zengarden-sample.css">css
file</A></SPAN></P></DIV>
<DIV id="preamble">
<H3><SPAN><xsl:value-of select="titre1"/></SPAN></H3>
<P class="p1"><SPAN>Littering a dark and dreary road lay the past relics of
browser-specific tags, incompatible <ACRONYM
title="Document Object Model">DOM</ACRONYM>s, and broken <ACRONYM
title="Cascading Style Sheets">CSS</ACRONYM> support.</SPAN></P>
<P class="p2"><SPAN>Today, we must clear the mind of past practices. Web
enlightenment has been achieved thanks to the tireless efforts of folk like the
<ACRONYM title="World Wide Web Consortium">W3C</ACRONYM>, <ACRONYM
title="Web Standards Project">WaSP</ACRONYM> and the major browser
creators.</SPAN></P>
<P class="p3"><SPAN>The css Zen Garden invites you to relax and meditate on the
important lessons of the masters. Begin to see with clarity. Learn to use the
(yet to be) time-honored techniques in new and invigorating fashion. Become one
with the web.</SPAN></P></DIV></DIV>
<DIV id="supportingText">
<DIV id="explanation">
<H3><SPAN><xsl:value-of select="titre2"/></SPAN></H3>
<P class="p1"><SPAN>There is clearly a need for <ACRONYM
title="Cascading Style Sheets">CSS</ACRONYM> to be taken seriously by graphic
artists. The Zen Garden aims to excite, inspire, and encourage participation. To
begin, view some of the existing designs in the list. Clicking on any one will
load the style sheet into this very page. The code remains the same, the only
thing that has changed is the external .css file. Yes, really.</SPAN></P>
<P class="p2"><SPAN><ACRONYM title="Cascading Style Sheets">CSS</ACRONYM> allows
complete and total control over the style of a hypertext document. The only way
this can be illustrated in a way that gets people excited is by demonstrating
what it can truly be, once the reins are placed in the hands of those able to
create beauty from structure. To date, most examples of neat tricks and hacks
have been demonstrated by structurists and coders. Designers have yet to make
their mark. This needs to change.</SPAN></P></DIV>
<DIV id="participation">
<H3><SPAN><xsl:value-of select="titre3"/></SPAN></H3>
<P class="p1"><SPAN>Graphic artists only please. You are modifying this page, so
strong <ACRONYM title="Cascading Style Sheets">CSS</ACRONYM> skills are
necessary, but the example files are commented well enough that even <ACRONYM
title="Cascading Style Sheets">CSS</ACRONYM> novices can use them as starting
points. Please see the <A title="A listing of CSS-related resources"
href="http://www.mezzoblue.com/zengarden/resources/"><ACRONYM
title="Cascading Style Sheets">CSS</ACRONYM> Resource Guide</A> for advanced
tutorials and tips on working with <ACRONYM
title="Cascading Style Sheets">CSS</ACRONYM>.</SPAN></P>
<P class="p2"><SPAN>You may modify the style sheet in any way you wish, but not
the <ACRONYM title="HyperText Markup Language">HTML</ACRONYM>. This may seem
daunting at first if you've never worked this way before, but follow the listed
links to learn more, and use the sample files as a guide.</SPAN></P>
<P class="p3"><SPAN>Download the sample <A
title="This page's source HTML code, not to be modified."
href="http://www.csszengarden.com/zengarden-sample.html">html file</A> and <A
title="This page's sample CSS, the file you may modify."
href="http://www.csszengarden.com/zengarden-sample.css">css file</A> to work on
a copy locally. Once you have completed your masterpiece (and please, don't
submit half-finished work) upload your .css file to a web server under your
control. <A title="Use the contact form to send us your CSS file"
href="http://www.mezzoblue.com/zengarden/submit/">Send us a link</A> to the file
and if we choose to use it, we will spider the associated images. Final
submissions will be placed on our server.</SPAN></P></DIV>
```

```
<DIV id="benefits">
<H3><SPAN><xsl:value-of select="titre4"/></SPAN></H3>
<P class="p1"><SPAN>Why participate? For recognition, inspiration, and a resource
we can all refer to when making the case for <ACRONYM
title="Cascading Style Sheets">CSS</ACRONYM>-based design. This is sorely
needed, even today. More and more major sites are taking the leap, but not
enough have. One day this gallery will be a historical curiosity; that day is
not today.</SPAN></P></DIV>
<DIV id="requirements">
<H3><SPAN><xsl:value-of select="titre5"/></SPAN></H3>
<P class="p1"><SPAN>We would like to see as much <ACRONYM
title="Cascading Style Sheets, version 1">CSS1</ACRONYM> as possible. <ACRONYM
title="Cascading Style Sheets, version 2">CSS2</ACRONYM> should be limited to
widely-supported elements only. The css Zen Garden is about functional,
practical <ACRONYM title="Cascading Style Sheets">CSS</ACRONYM> and not the
latest bleeding-edge tricks viewable by 2% of the browsing public. The only real
requirement we have is that your <ACRONYM
title="Cascading Style Sheets">CSS</ACRONYM> validates.</SPAN></P>
<P class="p2"><SPAN>Unfortunately, designing this way highlights the flaws in the
various implementations of <ACRONYM
title="Cascading Style Sheets">CSS</ACRONYM>. Different browsers display
differently, even completely valid <ACRONYM
title="Cascading Style Sheets">CSS</ACRONYM> at times, and this becomes
maddening when a fix for one leads to breakage in another. View the <A
title="A listing of CSS-related resources"
href="http://www.mezzoblue.com/zengarden/resources/">Resources</A> page for
information on some of the fixes available. Full browser compliance is still
sometimes a pipe dream, and we do not expect you to come up with pixel-perfect
code across every platform. But do test in as many as you can. If your design
doesn't work in at least IE5+/Win and Mozilla (run by over 90% of the
population), chances are we won't accept it.</SPAN></P>
<P class="p3"><SPAN>We ask that you submit original artwork. Please respect
copyright laws. Please keep objectionable material to a minimum; tasteful nudity
is acceptable, outright pornography will be rejected.</SPAN></P>
<P class="p4"><SPAN>This is a learning exercise as well as a demonstration. You
retain full copyright on your graphics (with limited exceptions, see <A
href="http://www.mezzoblue.com/zengarden/submit/guidelines/">submission
guidelines</A>), but we ask you release your <ACRONYM
title="Cascading Style Sheets">CSS</ACRONYM> under a Creative Commons license
identical to the <A title="View the Zen Garden's license information."
href="http://creativecommons.org/licenses/sa/1.0/">one on this site</A> so that
others may learn from your work.</SPAN></P>
<P class="p5"><SPAN>Bandwidth graciously donated by <A
href="http://www.dreamfirestudios.com/">DreamFire
Studios</A>.</SPAN></P></DIV>
<DIV id="footer"><A title="Check the validity of this site's XHTML"
href="http://validator.w3.org/check/referer">xhtml</A>  <A
title="Check the validity of this site's CSS"
href="http://jigsaw.w3.org/css-validator/check/referer">css</A>  <A
title="View details of the license of this site, courtesy of Creative Commons."
href="http://creativecommons.org/licenses/by-nc-sa/1.0/">cc</A>  <A
title="Check the accessibility of this site according to U.S. Section 508"
href="http://bobby.watchfire.com/bobby/bobbyServlet?URL=http%3A%2F%2Fwww.mezzoblue.com%2Fzengarden%2F
&amp;output=Submit&amp;gl=sec508&amp;test=">508</A>
<A
title="Check the accessibility of this site according to WAI Content Accessibility Guidelines 1"
href="http://bobby.watchfire.com/bobby/bobbyServlet?URL=http%3A%2F%2Fwww.mezzoblue.com%2Fzengarden%2F
&amp;output=Submit&amp;gl=wcag1-aaa&amp;test=">aaa</A>
</DIV></DIV>
<DIV id="linkList">
<DIV id="linkList2">
<DIV id="lselect">
<H3 class="select"><SPAN><xsl:value-of select="titre6"/></SPAN></H3>
<UL>
  <LI><A title="AccessKey: a" accessKey="a"
  href="http://www.csszengarden.com/?cssfile=/103/103.css&amp;page=0">Odyssey</A>
  by <A class="c" href="http://www.liquidarch.com/">Terrence Conley</A></LI>
  <LI><A title="AccessKey: b" accessKey="b"
  href="http://www.csszengarden.com/?cssfile=/102/102.css&amp;page=0">Revolution!</A>
```

```
 by <A class="c" href="http://www.monc.se/">David Hellsing</A></LI>
 <LI><A title="AccessKey: c" accessKey="c"
 href="http://www.csszengarden.com/?cssfile=/101/101.css&amp;page=0">punkass</A>
 by <A class="c" href="http://www.mikhel.com/">Mikhel Proulx</A></LI>
 <LI><A title="AccessKey: d" accessKey="d"
 href="http://www.csszengarden.com/?cssfile=/100/100.css&amp;page=0">15
 Petals</A> by <A class="c" href="http://www.meyerweb.com/">Eric Meyer &amp; Dave
 Shea</A></LI>
 <LI><A title="AccessKey: e" accessKey="e"
 href="http://www.csszengarden.com/?cssfile=/099/099.css&amp;page=0">Wiggles
 the Wonderworm</A> by <A class="c" href="http://www.make-believe.org/">Joseph
 Pearson</A></LI>
 <LI><A title="AccessKey: f" accessKey="f"
 href="http://www.csszengarden.com/?cssfile=/098/098.css&amp;page=0">Edo and
 Tokyo</A> by <A class="c" href="http://www.coutworks.com/">Daisuke Sato</A></LI>
 <LI><A title="AccessKey: g" accessKey="g"
 href="http://www.csszengarden.com/?cssfile=/097/097.css&amp;page=0">No
 Frontiers!</A> by <A class="c" href="http://hyperreal.info/bhang/">Michal
 Mokrzycki</A></LI>
 <LI><A title="AccessKey: h" accessKey="h"
 href="http://www.csszengarden.com/?cssfile=/096/096.css&amp;page=0">Japanese
 Garden</A> by <A class="c" href="http://www.jugglinglife.org/">Masanori
 Kawachi</A> </LI>
</UL></DIV>
<DIV id="larchives">
<H3 class="archives"><SPAN>Archives:</SPAN></H3>
<UL>
 <LI><A title="View next set of designs. AccessKey: n" accessKey="n"
 href="http://www.csszengarden.com/?cssfile=/001/001.css&amp;page=1"><SPAN
 class="accesskey">n</SPAN>ext designs »</A> </LI>
 <LI><A title="View every submission to the Zen Garden. AccessKey: w"
 accessKey="w" href="http://www.mezzoblue.com/zengarden/alldesigns/">Vie<SPAN
 class="accesskey">w</SPAN> All Designs</A> </LI></UL></DIV>
<DIV id="lresources">
<H3 class="resources"><SPAN>Resources:</SPAN></H3>
<UL>
 <LI><A
 title="View the source CSS file for the currently-viewed design, AccessKey: v"
 accessKey="v"
 href="css Zen Garden The Beauty in CSS Design_fichiers/001.css"><SPAN
 class="accesskey">V</SPAN>iew This Design's <ACRONYM
 title="Cascading Style Sheets">CSS</ACRONYM></A></LI>
 <LI><A
 title="Links to great sites with information on using CSS. AccessKey: r"
 accessKey="r" href="http://www.mezzoblue.com/zengarden/resources/"><ACRONYM
 title="Cascading Style Sheets">CSS</ACRONYM> <SPAN
 class="accesskey">R</SPAN>esources</A></LI>
 <LI><A
 title="A list of Frequently Asked Questions about the Zen Garden. AccessKey: q"
 accessKey="q" href="http://www.mezzoblue.com/zengarden/faq/"><ACRONYM
 title="Frequently Asked Questions">FA<SPAN
 class="accesskey">Q</SPAN></ACRONYM></A></LI>
 <LI><A title="Send in your own CSS file. AccessKey: s" accessKey="s"
 href="http://www.mezzoblue.com/zengarden/submit/"><SPAN
 class="accesskey">S</SPAN>ubmit a Design</A> </LI>
 <LI><A title="View translated versions of this page. AccessKey: t" accessKey="t"
 href="http://www.mezzoblue.com/zengarden/translations/"><SPAN
 class="accesskey">T</SPAN>ranslations</A> </LI></UL></DIV></DIV></DIV></DIV><!-- These extra divs/spans may
be used as catch-alls to add extra imagery. -->
<DIV id="extraDiv1"><SPAN></SPAN></DIV>
<DIV id="extraDiv2"><SPAN></SPAN></DIV>
<DIV id="extraDiv3"><SPAN></SPAN></DIV>
<DIV id="extraDiv4"><SPAN></SPAN></DIV>
<DIV id="extraDiv5"><SPAN></SPAN></DIV>
<DIV id="extraDiv6"><SPAN></SPAN></DIV>
</xsl:for-each>
</BODY></HTML>
</xsl:template>
</xsl:stylesheet>
```

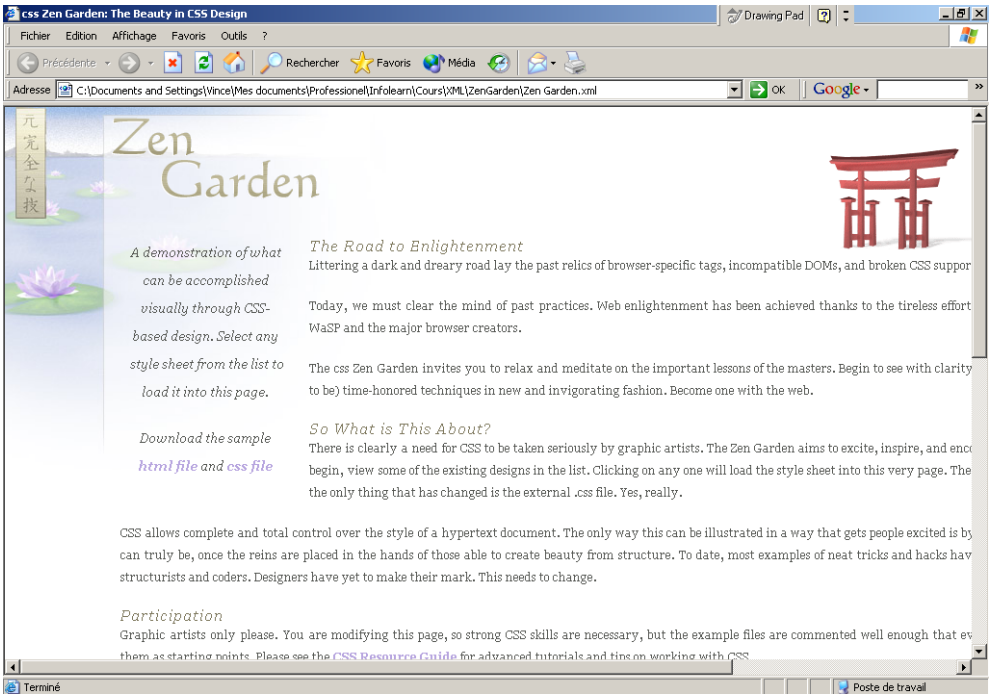
Vous remarquerez que nous faisons appel à un fichier CSS au tout début (mis en évidence dans le rectangle rouge transparent). Au fait, nous disposons de quatre de ces fichiers CSS qui sont trop longs pour être présentés ici. L'idée est la suivante:

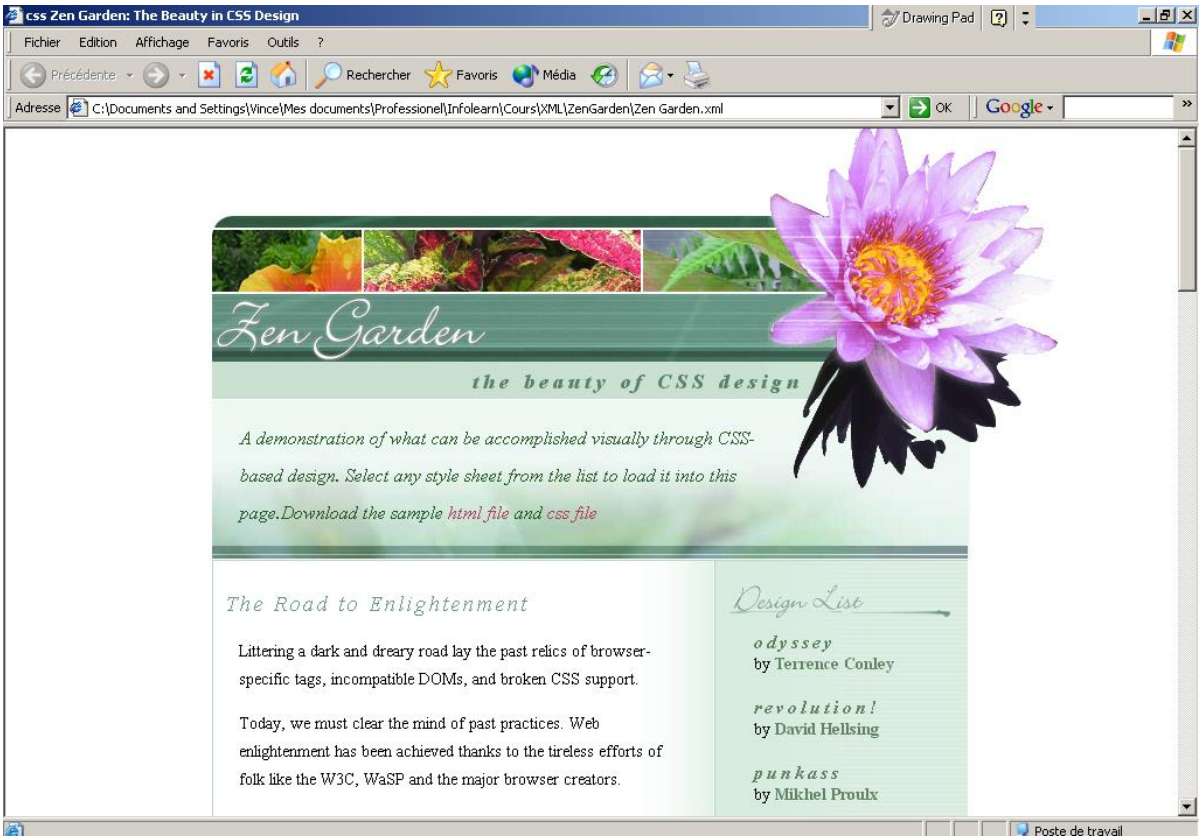
1. Nous considérons que vous êtes un spécialiste Web XHTML, XSL et CSS et que vous créez des modèles de pages avec des images et tout et tout pour les employés d'une entreprise (logiquement la création des modèles de documents et la mise en page de document ne fait pas partie du travail des employés administratifs ou cadres d'une entreprise… normalement… mais ça ils l'ont pas tous compris)

2. Dans le but d'optimiser la gestion des document types, votre entreprise vous demande alors de créer un système tel que les employés n'aient qu'à remplir le fichier XML contenant les textes bruts (ils ne doivent plus s'occuper de la mise en page et il leur suffit juste de connaître le bloc notes ou d'utiliser un logiciel de saisie qui génère automatiquement derrière un fichier XML → donc pas d'investissements aberrants dans des formations coûteuses sur MS Word et pas de perte de temps de travail sur du "n'importe quoi/n'importe comment" par les employés)

Bien évidemment, le résultat final se verra dans un navigateur Web (pour l'instant à ce niveau du discours) mais au fait ce qu'il faut savoir c'est que l'on peut faire des mises en page en qualité égales et même supérieures sur des pages web que par rapport à MS Word (et d'ailleurs les pages web sont plus légères…).

Eh bien! Voilà donc ce qu'on peut faire à partir du même fichier XML (dont on peut bien sûr changer les données) et en changeant seulement le type de fichier CSS appelé (dans l'ordre: 001.css, 002.css, 003.css, 004.css):


css Zen Garden: The Beauty in CSS Design
Drawing Pad
Fichier Edition Affichage Favoris Outils ?
Précédente
Rechercher
Favoris
Média
Adresse
C:\Documents and Settings\Vince\Mes documents\Professionel\Infolearn\Cours\XML\ZenGarden\Zen Garden.xml
OK
Google
Zen Garden
A demonstration of what can be accomplished visually through CSS-based design. Select any style sheet from the list to load it into this page.
Download the sample html file and css file
The Road to Enlightenment
Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, and broken CSS suppor
Today, we must clear the mind of past practices. Web enlightenment has been achieved thanks to the tireless effort
WaSP and the major browser creators.
The css Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity
to be) time-honored techniques in new and invigorating fashion. Become one with the web.
So What is This About?
There is clearly a need for CSS to be taken seriously by graphic artists. The Zen Garden aims to excite, inspire, and enc
begin, view some of the existing designs in the list. Clicking on any one will load the style sheet into this very page. The
the only thing that has changed is the external .css file. Yes, really.
CSS allows complete and total control over the style of a hypertext document. The only way this can be illustrated in a way that gets people excited is by
can truly be, once the reins are placed in the hands of those able to create beauty from structure. To date, most examples of neat tricks and hacks hav
structurists and coders. Designers have yet to make their mark. This needs to change.
Participation
Graphic artists only please. You are modifying this page, so strong CSS skills are necessary, but the example files are commented well enough that ev
them as starting points. Please see the CSS Resource Guide for advanced tutorials and tips on working with CSS
Terminé
Poste de travail



Et bien demandez à votre collègue de faire ces 4 documents sous MS Word avec le même texte de manière classique… vous pouvez attendre longtemps… alors qu'avec XML, XSL, XHTML, MS Office 2003 et une bonne stratégie d'entreprise pour les documents types, on gagne un temps phénoménal, on optimise donc les processus de travail, on gagne en coût de

formation et on produit des documents d'une qualité professionnelle à tous les niveaux (certains employés n'ayant dès lors qu'à saisir du texte dans le fichier XML pour remplir la page sans avoir à s'occuper de la mise en page).

Donc les utilisateurs lambda ne font plus qu'une chose pour les documents type: DE LA SAISIE !!! (enfin c'est pas trop tôt…)

Nous verrons par la suite que un avantage monstrueux c'est que le contenu des fichiers XML peuvent êtres importés dans des fichiers MS Access et MS Excel et ainsi toute entreprise obtient un GED de moyenne ou bonne qualité a très faible coût (à la portée de tout informaticien).

PS: les exemples ont été inspirés (j'ai du changer le HTML en XHTML et sortir les titres dans un fichier XML) du site www.csszengarden.com (le site de référence pour voir la puissance des styles CSS sur le web).

Remarque: au fait, comme nous le verrons lors de notre étude de MS Word 2003 et XML, ce premier s'oriente gentiment dans la conception de documents web avec du CSS… eh oui…. c'est ça l'avenir… pour les pros du traitement de texte: faire tout ce que nous avons fait jusqu'à maintenant mais… dans MS Word !!!! Mais nous verrons cela en détail plus loin.

5.12 XSL: Espaces de noms (xmlns)

Le caractères extensible du langage XML offre d'énormes possibilités aux concepteurs de documents, mais pose aussi un certain nombre de problèmes. En effet, en fonction de l'auteur ou du contexte du document, la signification d'un élément peut différer. En d'autres termes, il est tout à fait possible que deux développeurs choisissent le même nom pour leur balise sans toutefois les utiliser de la même manière.

Toutefois, XML a été créé pour apporter un contexte à l'information; il serait paradoxal d'avoir à se servir de cette dernière pour comprendre les balises ! C'est l'environnement lexical qui permet de comprendre de quel type de pièce il s'agit. L'équivalent informatique de ces champs lexicaux se nomme "espace de noms" (namespace). Pour désigner un espace de nom, nous utilisons une URI qui identifie le namespace.

Un URI (Uniform Resource Identifier) se compose d'une chaîne de caractères unique qui définit l'emplacement d'une ressource (le fichier XML lui-même souvent!). Attention: il s'agit souvent d'un identifiant symbolique. En d'autres termes, il est possible que la ressource ne se trouve pas vraiment à l'endroit indiquée par l'URI.

Reprenons un exemple déjà traité lors de notre introduction au XSL pour les namespaces en l'adaptant un tant soit peu:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="OrderByEspNoms.xsl"?>
<compilation xmlns:cmpl="http://www.sciences.ch">
    <cmpl:mp3>
        <titre>Foule sentimentale</titre>
        <artiste>Alain Souchon</artiste>
    </cmpl:mp3>
    <cmpl:mp3>
        <titre>Solaar pleure</titre>
        <artiste>MC Solaar</artiste>
    </cmpl:mp3>
    <cmpl:mp3>
        <titre>Le baiser</titre>
```

```
        <artiste>Alain Souchon</artiste>
    </cmpl:mp3>
    <cmpl:mp3>
        <titre>Pourtant</titre>
        <artiste>Vanessa Paradis</artiste>
    </cmpl:mp3>
    <cmpl:mp3>
        <titre>Chambre avec vue</titre>
        <artiste>Henri Salvador</artiste>
    </cmpl:mp3>
</compilation>
```

Nous définissons ci-dessus le namespace cmpl (abréviation de "compilation") afin de distinguer le type de données. Mais rien ne nous empêche de créer autant de xmlns que nous voulons. Enfin, le fichier XSL rattaché se voit modifié un petit peu tel que ci-dessous:

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

Le fichier XSD correspondant étant:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified"
attributeFormDefault="unqualified" targetNamespace="uri:MonNamespace ">
<!-- Si on pose le elementFormDefault sur qualified cela signifie que tous les champs doivent appartenir au namespace,
sinon s'il est égal à unqualified, seulement l'élément racine appartiendra au namespace alors que les autres noeuds
appartiendront à un namespace vide-->
    <xs:element name="compilation">
        <xs:annotation>
            <xs:documentation>Comment describing your root element</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="mp3" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="titre"/>
                            <xs:element name="artiste"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

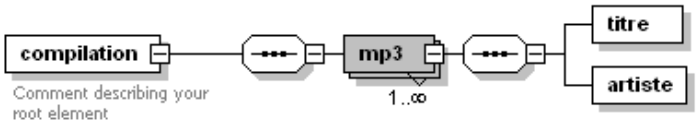
1. Remarquez d'abord le targetNamespace="MonNameSpace", il devrait faire référence à une ressource en ligne expliquant les tenants et aboutissant de votre espace de noms au même titre que toutes les url utilisées actuellement dans les namespaces.

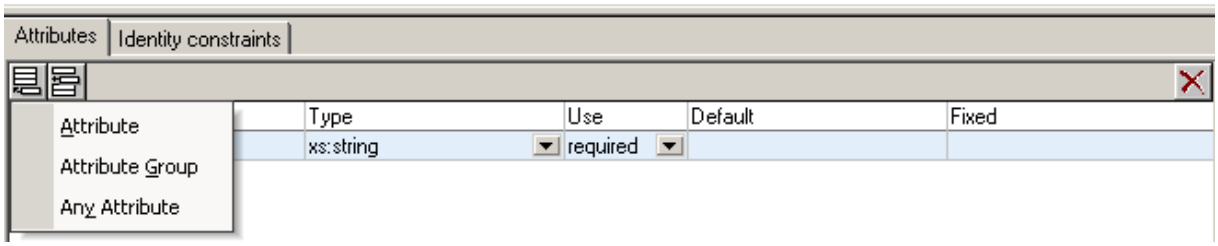
2. Maintenant remarquez surtout (nous l'avons changé par rapport au défaut que propose XMLSpy) le elementFormDefault="unqualified" qui définit simplement le fait que les noeuds filles du nœud racine ne doivent pas utiliser l'espace de noms du nœud racine (soit un espace de noms vide dans notre exemple). Ainsi, toujous avec notre exemple, le fichier XML suivant ne serait actuellement pas valide:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<cmpl:compilation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:cmpl="uri:MonNamespace"
xsi:noNamespaceSchemaLocation="Schema.xsd">
    <cmpl:mp3>
        <cmpl:titre>Foule sentimentale</cmpl:titre>
        <cmpl:artiste>Alain Souchon</cmpl:artiste>
    </cmpl:mp3>
    <cmpl:mp3>
        <cmpl:titre>Solaar pleure</cmpl:titre>
        <cmpl:artiste>MC Solaar</cmpl:artiste>
    </cmpl:mp3>
    <cmpl:mp3>
        <cmpl:titre>Le baiser</cmpl:titre>
        <cmpl:artiste>Alain Souchon</cmpl:artiste>
    </cmpl:mp3>
    <cmpl:mp3>
        <cmpl:titre>Pourtant</cmpl:titre>
        <cmpl:artiste>Vanessa Paradis</cmpl:artiste>
    </cmpl:mp3>
    <cmpl:mp3>
        <cmpl:titre>Chambre avec vue</cmpl:titre>
        <cmpl:artiste>Henri Salvador</cmpl:artiste>
    </cmpl:mp3>
</cmpl:compilation>
```

alors que si nous mettions dans le fichier XSD elementFormDefault="qualified" le fichier sera reconnu comme valide.

Maintenant ajoutez un attribut au nœud *mp3* de type *style* tel que ci-dessous à l'aide de XMLSpy:





avec:

ce qui donne dans le code XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="MonNameSpace" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <!-- Si on pose le elementFormDefault sur qualified cela signifie que tous les champs doivent appartenir au
namespace, sinon si il est égal à unqualified, seulement l'élément racine appartiendra au namespace alors que les
autres noeuds appartiendront à un namespace vide-->
    <xs:element name="compilation">
        <xs:annotation>
            <xs:documentation>Comment describing your root element</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="mp3" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="titre"/>
                            <xs:element name="artiste"/>
                        </xs:sequence>
                        <xs:attribute name="style" type="xs:string" use="required"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Maintenant changez votre fichier XML comme suit:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<cmpl:compilation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:cmpl="MonNameSpace"
xsi:noNamespaceSchemaLocation="Namespace.xsd">
    <cmpl:mp3 style="Variétés">
```

```
        <cmpl:titre>Foule sentimentale</cmpl:titre>
        <cmpl:artiste>Alain Souchon</cmpl:artiste>
    </cmpl:mp3>
    <cmpl:mp3 style="Rap">
        <cmpl:titre>Solaar pleure</cmpl:titre>
        <cmpl:artiste>MC Solaar</cmpl:artiste>
    </cmpl:mp3>
    <cmpl:mp3 style="Variétés">
        <cmpl:titre>Le baiser</cmpl:titre>
        <cmpl:artiste>Alain Souchon</cmpl:artiste>
    </cmpl:mp3>
    <cmpl:mp3 style="Variétés">
        <cmpl:titre>Pourtant</cmpl:titre>
        <cmpl:artiste>Vanessa Paradis</cmpl:artiste>
    </cmpl:mp3>
    <cmpl:mp3 style="Musique du monde">
        <cmpl:titre>Chambre avec vue</cmpl:titre>
        <cmpl:artiste>Henri Salvador</cmpl:artiste>
    </cmpl:mp3>
</cmpl:compilation>
```

Le document est conforme et valide avec notre XSD. Mais mettez dans le XSD attributeFormDefault="qualified" votre fichier xml sera alors toujours conforme mais plus valide. Il faudra (comme vous l'avez certainement deviné…) changer votre fichier XML comme ci-dessous pour qu'il soit à nouveau valide:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<cmpl:compilation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:cmpl="MonNameSpace"
xsi:noNamespaceSchemaLocation="Namespace.xsd">
    <cmpl:mp3 cmpl:style="Variétés">
        <cmpl:titre>Foule sentimentale</cmpl:titre>
        <cmpl:artiste>Alain Souchon</cmpl:artiste>
    </cmpl:mp3>
    <cmpl:mp3 cmpl:style="Rap">
        <cmpl:titre>Solaar pleure</cmpl:titre>
        <cmpl:artiste>MC Solaar</cmpl:artiste>
    </cmpl:mp3>
    <cmpl:mp3 cmpl:style="Variétés">
        <cmpl:titre>Le baiser</cmpl:titre>
        <cmpl:artiste>Alain Souchon</cmpl:artiste>
    </cmpl:mp3>
    <cmpl:mp3 cmpl:style="Variétés">
        <cmpl:titre>Pourtant</cmpl:titre>
        <cmpl:artiste>Vanessa Paradis</cmpl:artiste>
    </cmpl:mp3>
    <cmpl:mp3 cmpl:style="Musique du monde">
        <cmpl:titre>Chambre avec vue</cmpl:titre>
        <cmpl:artiste>Henri Salvador</cmpl:artiste>
    </cmpl:mp3>
</cmpl:compilation>
```

6. Transformer XML en XML avec XSL

Certaines entreprises travaillant avec des bases de données particulières ont le besoin de travailler avec du XSL pour exporter certaines données d'un flux XML en XML.

Ceci n'est pas bien compliqué car utilise des techniques qui nous sont en très grande partie déjà connues. Ce qu'il faut simplement c'est d'un moteur capable d'effectuer l'opération de transformation. Certains progiciels en sont fournis comme c'est le cas de XMLSpy

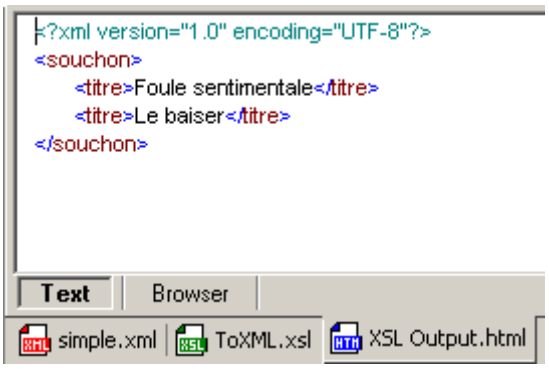
Voyons cela sur un exemple simple. Soit le fichier XML suivant (remarquez la référence à ToXML.xsl):

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="ToXML.xsl"?>
<compilation>
    <mp3>
        <titre>Foule sentimentale</titre>
        <artiste>Alain Souchon</artiste>
    </mp3>
    <mp3>
        <titre>Solaar pleure</titre>
        <artiste>MC Solaar</artiste>
    </mp3>
    <mp3>
        <titre>Le baiser</titre>
        <artiste>Alain Souchon</artiste>
    </mp3>
    <mp3>
        <titre>Pourtant</titre>
        <artiste>Vanessa Paradis</artiste>
    </mp3>
    <mp3>
        <titre>Chambre avec vue</titre>
        <artiste>Henri Salvador</artiste>
    </mp3>
</compilation>
```

Nous aimerions en extraire un fichier XML contenant seulement les titres d'Alain Souchon. Le code XSL permettant de faire une telle chose pourrait s'écrire sous la forme suivante:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" omit-xml-declaration="no" indent="yes" encoding="UTF-8"/>
<xsl:template match="/">
    <souchon>
        <xsl:for-each select="compilation/mp3[artiste='Alain Souchon']">
            <titre>
                <xsl:value-of select="titre"/>
            </titre>
        </xsl:for-each>
    </souchon>
</xsl:template>

</xsl:stylesheet>
```

Ensuite, dans XMLSpy il suffit de cliquer sur le bouton  pour obtenir un fichier nommé *XML Output.htm* dont le contenu texte est:


<?xml version="1.0" encoding="UTF-8"?>
<souchon>
<titre>Foule sentimentale</titre>
<titre>Le baiser</titre>
</souchon>
Text
Browser
simple.xml
ToXML.xsl
XSL Output.html

7. Transformer XML en SQL avec XSL

Il arrive parfois qu'il ne soit pas possible d'importer des fichiers XML dans des systèmes de bases de données ou des tableurs. Une solution possible est alors de transformer le contenu d'un fichier XML à l'aide du XSL en une commande SQL de type INSERT INTO (voir mon support sur MS Office Access). POur l'exemple, partons du fichier XML suivant:

```
<?xml version="1.0" encoding="UTF-8"?>
- <diplomes>
  - <parametres>
      <type-export>complet</type-export>
      <horodatage>31/08/2010 22:01:07</horodatage>
    </parametres>
  - <fiche-diplome>
      <type-intervention>C</type-intervention>
      <code-diplome>1874</code-diplome>
      <code-type-diplome>105</code-type-diplome>
      <intitule-type-diplome>Mastère spécialisé</intitule-type-diplome>
      <code-niveau>1</code-niveau>
      <code-niveau-europeen/>
      <code-scolarite/>
      <code-sise>9730703</code-sise>
      <code-onisep/>
      <code-onisep-ideo/>
      <code-specificite-ideo/>
      <code-cqp/>
      <code-ministere-emploi/>
      <code-afpa/>
      <code-nsf>116</code-nsf>
      <code-lettre-nsf/>
      <code-carif-07/>
      <code-carif-26/>
      <inscrit-rncp>2</inscrit-rncp>
      <type-sigle>Mastère spécialisé</type-sigle>
      <type-complet>Mastère spécialisé</type-complet>
      <intitule-diplome>Mastère spécialisé mastère spécialisé chimie 'ts' ttt</intitule-diplome>
      <dominante>chimie</dominante>
      <mention/>
      <specialite/>
      <intitule-sigle/>
      <code-niveau-entree/>
      <accessibilite-fi/>
      <accessibilite-ca>2</accessibilite-ca>
      <accessibilite-fc>2</accessibilite-fc>
      <accessibilite-cp>2</accessibilite-cp>
      <accessibilite-vae>1</accessibilite-vae>
      <accessibilite-ind>2</accessibilite-ind>
      <accessibilite-uc/>
      <texte-chapo/>
      <texte-objectif/>
      <texte-programme/>
      <texte-admission/>
```

```
        <texte-admission/>
        <texte-poursuite/>
        <texte-debouches/>
        <texte-source/>
        <valideur/>
        <annee-premiere-session>2002</annee-premiere-session>
        <annee-derniere-session/>
        <etat>3</etat>
        <date-creation>11/02/2004</date-creation>
        <date-maj>11/02/2004</date-maj>
        <validation-interne/>
        <validation-alfa/>
      - <modalites-diplomes>
          - <modalite>
              <code-type-modalite>2</code-type-modalite>
              <code-modalite>5</code-modalite>
              <libelle>Label de la Conférence des grandes écoles (CGE)</libelle>
            </modalite>
        </modalites-diplomes>
        <jos> </jos>
        <modules> </modules>
        <liens-diplomes-formacodes> </liens-diplomes-formacodes>
        <liens-diplomes-gfe> </liens-diplomes-gfe>
        <liens-diplomes-romes> </liens-diplomes-romes>
        <liens-diplomes-fap> </liens-diplomes-fap>
      - <liens-diplomes-validations>
          - <lien-diplomes-validations>
              <code-validation>1015</code-validation>
              <nom-validation>Diplôme de la recherche</nom-validation>
              <certificateur>1</certificateur>
              <valideur/>
              <code-habilitation/>
              <code-rncp/>
              <duree-habilitation/>
              <date-ouverture>00/00/0000</date-ouverture>
              <date-fermeture>00/00/0000</date-fermeture>
              <validite-actuelle/>
              <date-maj>01/01/2007</date-maj>
            </lien-diplomes-validations>
        </liens-diplomes-validations>
        <historiques-diplomes> </historiques-diplomes>
    </fiche-diplome>
</diplomes>
```

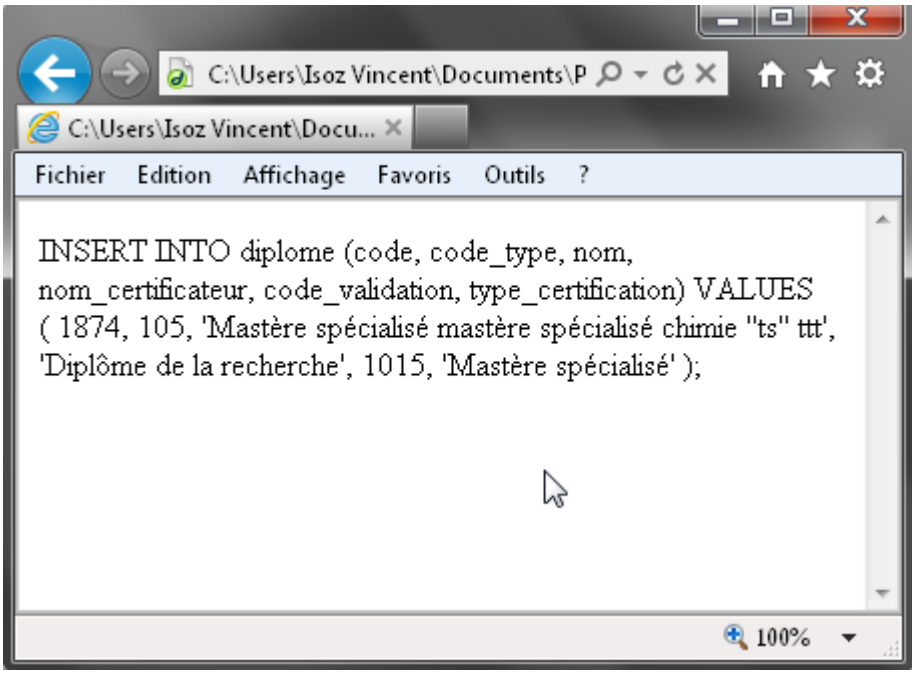
et le fichier XLS (certif.xsl) suivant:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:param name="apostrophe">'</xsl:param>
  <xsl:template name="cleanQuote">
    <xsl:param name="string"/>
    <xsl:if test="contains($string, $apostrophe)">
      <xsl:value-of select="substring-before($string, $apostrophe)"/>
      ''
      <xsl:call-template name="cleanQuote">
        <xsl:with-param name="string">
          <xsl:value-of select="substring-after($string, $apostrophe)"/>
        </xsl:with-param>
      </xsl:call-template>
    </xsl:if>
    <xsl:if test="not(contains($string, $apostrophe))">
      <xsl:value-of select="$string"/>
    </xsl:if>
  </xsl:template>
  <xsl:template match="diplomes/parametres"/>
  <xsl:template match="diplomes/fiche-diplome">
    <xsl:text> INSERT INTO diplome (code, code_type, nom, nom_certificateur, code_validation, type_certification) VALUES ( </xsl:text>
    <xsl:value-of select="code-diplome"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="code-type-diplome"/>
    <xsl:text>, '</xsl:text>
    <xsl:call-template name="cleanQuote">
      <xsl:with-param name="string">
        <xsl:value-of select="intitule-diplome"/>
      </xsl:with-param>
    </xsl:call-template>
    <xsl:text>', '</xsl:text>
    <xsl:call-template name="cleanQuote">
      <xsl:with-param name="string">
        <xsl:value-of select="liens-diplomes-validations/lien-diplomes-validations/nom-validation"/>
      </xsl:with-param>
    </xsl:call-template>
    <xsl:text>', </xsl:text>
    <xsl:value-of select="liens-diplomes-validations/lien-diplomes-validations/code-validation"/>
    <xsl:text>, '</xsl:text>
    <xsl:call-template name="cleanQuote">
      <xsl:with-param name="string">
        <xsl:value-of select="intitule-type-diplome"/>
      </xsl:with-param>
    </xsl:call-template>
    <xsl:text>' </xsl:text>
    <xsl:text>);</xsl:text>

    <br/>
  </xsl:template>
</xsl:stylesheet>
```

Nous ajoutons dans le fichier XML la ligne suivante:

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

Et si nous ouvrons le fichier XML dans un navigateur (typiquement Internet Explorer), nous aurons (attention l'exemple est petit car il n'y a qu'un seul diplôme dans le fichier XML):


C:\Users\Isoz Vincent\Documents\P
C:\Users\Isoz Vincent\Docu...
Fichier Edition Affichage Favoris Outils ?
INSERT INTO diplome (code, code_type, nom, nom_certificateur, code_validation, type_certification) VALUES (1874, 105, 'Mastère spécialisé mastère spécialisé chimie "ts" ttt', 'Diplôme de la recherche', 1015, 'Mastère spécialisé');
100%

8. Transformer XML en SQL avec XSL

Partons du fichier suivant:

9. R.S.S.

Le R.S.S. (Ressource (Description Framework) Site Summary) est un contenu XML utilisé en standard de par le monde pour échanger des "news" de tout type sur le web mais sous une norme bien définie et ainsi utilisable par toutes et par tous gratuitement ou non par un traitement approprié des données (XSL, PHP, ASP, .Net, Java, MS Word, MS Excel, MS Access,…).

Voici un exemple simple de fichier RSS (extrait tiré de mon site perso. …) dont les balises sont conformes à la norme internationale (j'ai cependant ajouté les balises <auteur> et <mois> pour un usage personnel).

Attention cet exemple ne marche que si les fichiers sont sur un serveur web local ou distant ou que dans XMLSPy, une fois le fichier XML ayant la référence au fichier XSL vous cliquiez sur  .

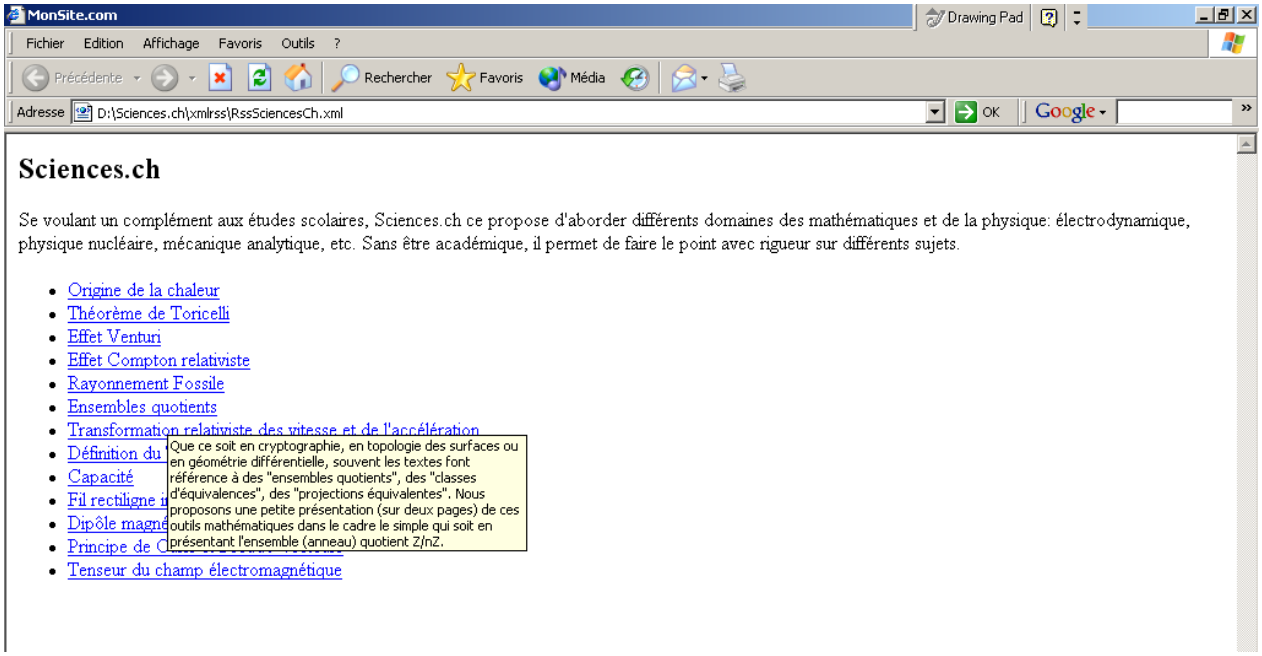
```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml:stylesheet type="text/xsl" href="RSStyleSheet.xsl"?>
<rss version="2.0">
    <channel>
        <title>Sciences.ch</title>
        <link>http://www.sciences.ch/xmlrss/index.php</link>
        <category>Mathématique, Physique théorique</category>
        <description>
Se voulant un complément aux études scolaires, Sciences.ch ce propose d'aborder différents domaines des mathématiques et de la physique: électrodynamique, physique nucléaire, mécanique analytique, etc. Sans être académique, il permet de faire le point avec rigueur sur différents sujets.
        </description>
        <language>fr</language>
        <copyright>Concept, Design, Contenu, 2002-2004 Sciences.ch. Tous droits résérvés</copyright>
        <image>
            <title>Sciences.ch</title>
            <url>http://www.sciences.ch/images/bansciencech.gif</url>
            <link>http://www.sciences.ch</link>
            <width>88</width>
            <height>31</height>
            <description>Au coeur de la Science!</description>
        </image>
        <item>
            <title>Origine de la chaleur</title>
            <description>Qu'est-ce que la chaleur ? Voici une question que se posent nombre d'étudiants et parfois suffisamment longtemps pour qu'ils n'obtiennent jamais la réponse même lors de leur cursus scolaire universitaire. Ne pas comprendre ce qu'est la chaleur est nous le savons, une source énorme de confusion et de difficulté de compréhension des conceptions fondamentaux de la thermodynamique (premier principe). Sciences.ch propose donc la une explication microscopique de la chaleur basée sur la mécanique statistique démontrant que cette première n'est qu'au fait qu'un changement de la distribution énergétique des micro-états vers de plus hautes ou plus faible valeurs. Il convient peut-être de se demander s'il ne faudrait pas introduire la thermodynamique seulement et seulement après avoir étudié les bases de la mécanique statistique  (étant donné les problèmes qu'ont les étudiants en thermodynamique) qui permettent d'accéder à la démonstration que nous proposons.</description>
            <link>http://www.sciences.ch/htmlfr/mecanique/mecanthermodyn01.php#chaleur</link>
            <pubDate>Je, 27 Mai 2004 11:50:01 GMT</pubDate>
        </item>
        <item>
            <title>Théorème de Toricelli</title>
            <description>Dans le cadre de la mécanique des milieux continus (M.M.C), nous pouvons nous poser un très grand nombre de question relativement au fonctionnement de nos appareil ménagers faisant usage de fluides à l'état liquide dont la manière de déterminer la vitesse d'écoulement d'un liquide sortant d'un récipient dans des cas où les écoules sont laminaires (non turbelements). C'est un cas classique d'étude dans les petites écoles. Nous proposons une démonstration de la relation connue dans les petites classes démontrée à partir de l'équation de Bernoulli (elle même tirée de la démonstration des équations de Navier-Stokes)</description>
            <link>http://www.sciences.ch/htmlfr/mecanique/mecanfluides01.php#toricelli</link>
        </item>
```

```
    </channel>
</rss>
```

et son fichier interne XSL correspondant (on ne peut plus simple):

```
<?xml version="1.0" encoding="iso-8859-1"?>
 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"/>
  <xsl:template match="/rss" >
   <html xml:lang="fr">
    <head>
     <title>MonSite.com</title>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    </head>
    <body>
     <xsl:for-each select="channel">
      <div class="channel">
       <h2><xsl:value-of select="title" /></h2>
       <p><xsl:value-of select="description" /></p>
       <ul>
        <xsl:for-each select="item">
         <li><a href="{link}" title="{description}"><xsl:value-of select="title" /></a></li>
        </xsl:for-each>
       </ul>
      </div>
     </xsl:for-each>
    </body>
   </html>
  </xsl:template>
 </xsl:stylesheet>
```

Pour obtenir:



Evidemment cette méthode fonctionne pour l'instant si le XML et XSL sont tous les deux sur le même serveur mais nous allons voir qu'il n'est pas bien dur de faire de même lorsque le fichier XML/RSS se trouve sur un serveur distant (voir la partie traitant du PHP et de l'ASP avec XML).

Remarque: on peut trouver de nombreux logiciels de lecteurs de flux gratuits sur Internet que ce soit sur PC (comme l'excellent *Newsvore* disponible sur www.labo-microsoft.com) ou pour Pocket PC (voir le site www.handango.com).

Citons pour les francophones aussi *AlertInfo* disponible à l'adresse suivante:

http://www.geste.fr/alertinfo/home.html

10. MS Office 2003

Le problème des grandes entreprises en ce début de 21ème siècle explique en partie la stratégie de Microsoft d'orienter sa suite bureautique vers le XML et donc par extension…au Web. Voici quels sont ces problèmes majeurs rencontrés par les grandes entreprises:

1. Normaliser les processus d'usage de l'informatique par ses employés

2. Réduire les coûts de formation à des logiciels complexes de traitement de l'information

3. Ramener chaque employé à se focaliser sur son travail et non sur les aléas de mise en page ou d'esthétisme de ses documents par l'application de modèles facilement utilisables et automatisés.

4. Centraliser la documentation (G.E.D. – Gestion Electronique de Documentation ce que va faire la prochaine version de Windows Server qui suit la 2003)

5. S'assurer d'un accès à des archives toujours fonctionnelles même dans 50 ans

6. Capacité à échanger des documents légers entre employés, serveurs, pages web, logiciels, etc. sous une forme commune et standardisée et compatibles avec un maximum de systèmes

7. et encore beaucoup d'autres (la liste est longue en ce qui concerne le mauvais usage des logiciels de la suite MS Office… et du reste… croyez-moi…)

C'est à tous ces niveaux qu'intervient le XML. Le mieux pour le comprendre mieux ce serait suivre un séminaire sur le sujet (…n'hésitez pas à me le faire savoir…) car il y a tellement de choses à dire qu'on pourrait écrire un livre sur les objectifs et les avantages (je n'ai pas encore trouvé de désavantages pour l'instant…).

Mais voyons étapes par étapes les possibilités de base du XML de la suite MS Office 2003 et espérant que l'esprit brillant qui lit ces lignes (…) saura en capter le potentiel.

Remarques:

R1. Comme nous en avons déjà fait mention, ce support ne s'adresse pas à des informaticien de métier. Ainsi, nous ne traiterons pas des nouveautés tels que le développement de Smart-Tags, Smart-Documents ou Webservices (InfoPath) avec .Net. Bien que cela soit fort passionnant, une grande quantité de sites y font déjà référence et les professionnels de la branche savent très bien s'accommoder de ce genre de supports techniques.

R2. Les outils de base de la suite MS Office 2003 ne gèrent pas les "attributs" des nœuds XML mis à part en passant par le VBA.

R3. MS Publisher 2003, MS OneNote 2003, MS Business Contact Manager ne proposent pas d'outils autres que la programmation pour traiter le XML. Nous n'aborderons donc pas logiciels.

10.1 MS Excel 2003

L'avantage de pouvoir travailler avec l'XML comme conteneur de données dans MS Excel est bien sûr la légèreté de ce format, sa compatibilité avec quasiment tous les systèmes existants et sa rigueur de traitement et sa flexibilité à être traiter dans n'importe quel logiciel compatible XML. Plus nous avancerons dans notre présentation de l'XML avec MS Office 2003 dans ce document, plus vous en comprendrez l'étendue.

Pour travailler avec le XML de manière professionnelle dans MS Excel 2003 il faut avant tout avoir un schéma de validation XSD des fichiers XML qu'utiliseront les employés de l'entreprise.

Considérons ainsi le fichier XML suivant dans un premier temps:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<NewDataSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ventes.xsd">
    <ventes>
        <au_id>172-32-1176</au_id>
        <au_lname>Einstein</au_lname>
        <au_fname>Albert</au_fname>
        <phone>408 496-723</phone>
        <address>10932 Bigge Rd.</address>
        <city>Menlo Park</city>
        <state>CA</state>
        <zip>94025</zip>
        <contract>true</contract>
        <ventes>600</ventes>
    </ventes>
    <ventes>
        <au_id>172-32-1233</au_id>
        <au_lname>Planck</au_lname>
        <au_fname>Max</au_fname>
        <phone>408 434-543</phone>
        <address>4304. Hollywood St.</address>
        <city>Holl. Bvd.</city>
        <state>CA</state>
        <zip>3454</zip>
        <contract>false</contract>
        <ventes>300</ventes>
    </ventes>
</NewDataSet>
```

C'est un simple fichier XML il n'y a rien à y redire. Mais il nous faut maintenant créer le fichier XSD correspondant ? Comment faire lorsqu'on a pas le temps mais les moyens (financiers) ?

Solution: acheter XMLSpy ! (ne vous en faites pas je suis pas revendeur…)… et là c'est du gâteau. Il suffit de faire des petits dessins pour que le schéma XSD se fasse tout seul.

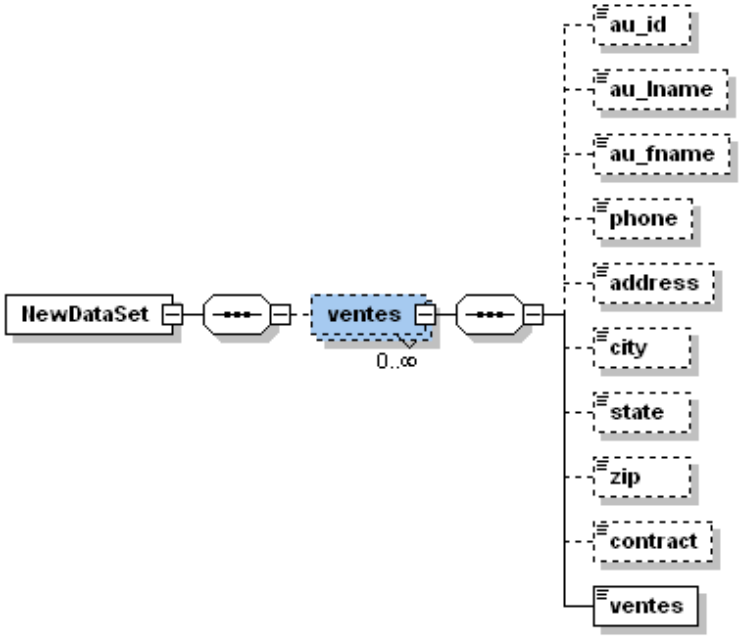
D'abord histoire de vous dégoûter un peu, voici le fichier XSD péniblement (supposé) écrit à la main:

```
<?xml version="1.0" standalone="yes"?>
<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by bob denard (Lost Paradise Inc.) -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xsd:element name="NewDataSet">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="ventes" minOccurs="0" maxOccurs="unbounded">
```

```
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="au_id" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="au_lname" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="au_fname" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="phone" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="address" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="city" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="state" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="zip" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="contract" type="xsd:boolean" minOccurs="0"/>
                            <xsd:element name="ventes" type="xsd:integer"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

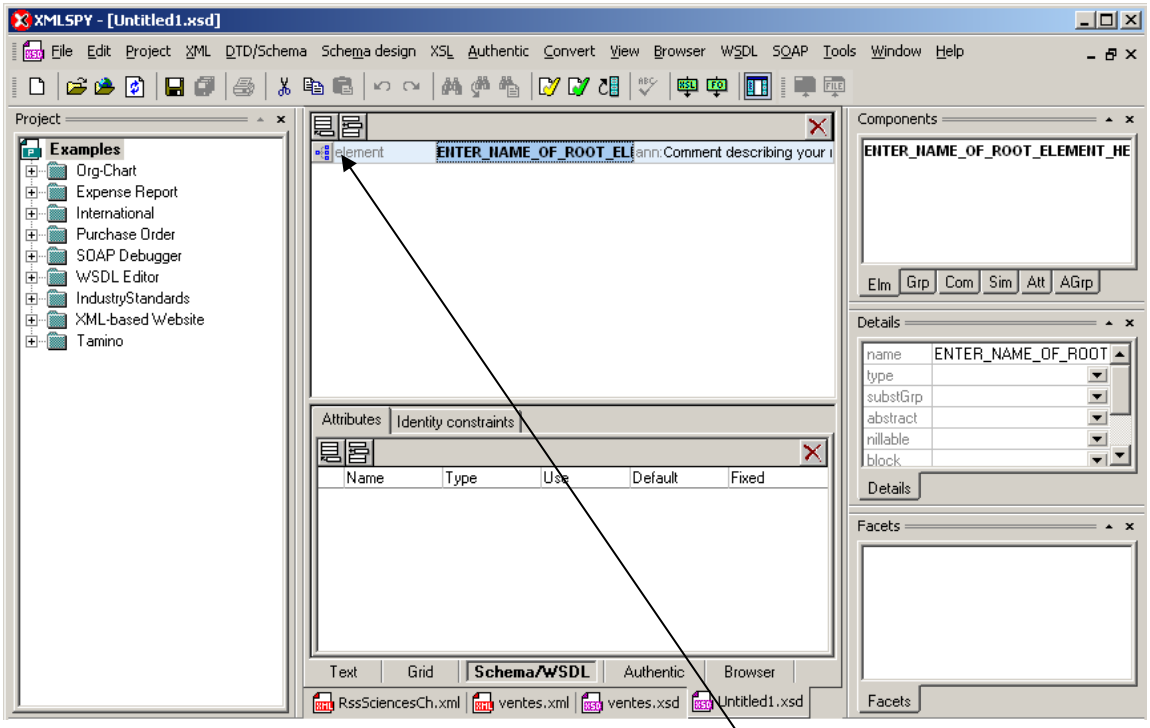
Pas franchement motivant non ?

Et voilà son équivalent sous forme graphique finale dans XMLSpy:



Comment faire ceci ? Et bien voici la marche à suivre:

Dans XMLSpy aller dans *File/New* et choisissez le type de fichier "*XSD*" (voir page suivante):

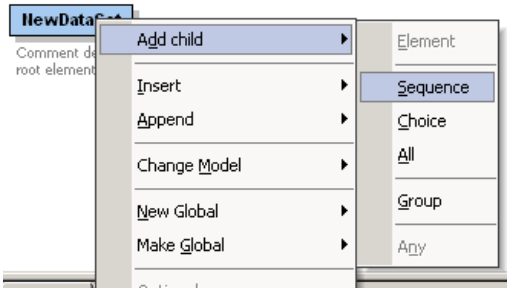


Cliquez sur l'onglet *Schema/WSDL* et ensuite sur le bouton: 

Ceci apparaîtra alors:

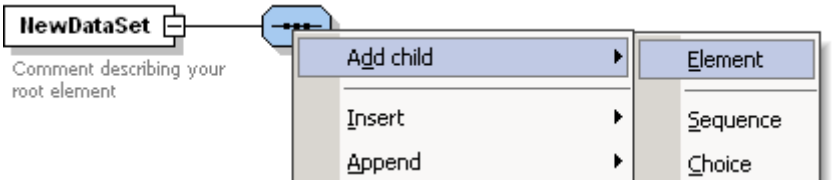
eh bien il suffit de faire ce qu'ils disent. On fait un double clique sur la petite boîte bleue et on écrire par rapport à notre fichier XML *NewDataSet*:

Ensuite, dans notre fichier XML nous voyons qu'il y a plusieurs nœuds (une "séquence" en d'autres termes) enfants nommés *ventes*. Pour ajouter cela au schéma WSDL faisons l'opération suivante (petit bouton droit sur la case bleue):

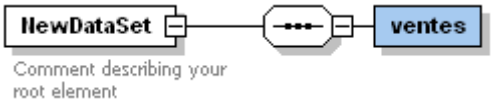


et voilà ce qui apparaît:

et puis ? Ben… où est *ventes* ? Alors à nouveau bouton droit:

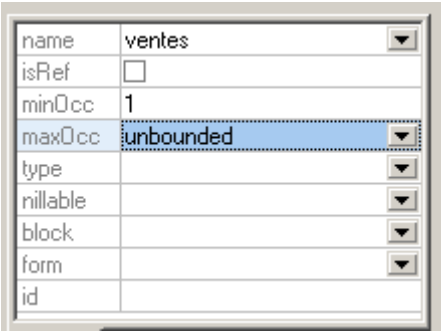


et un petit double cliquer pour renommer cela *ventes* tel que:

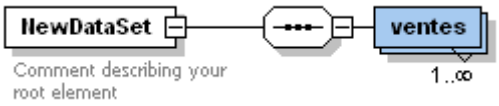


Mais… mais… certes il y a une séquence de *ventes* mais combien au minimum autorisés et combien au maximum ?

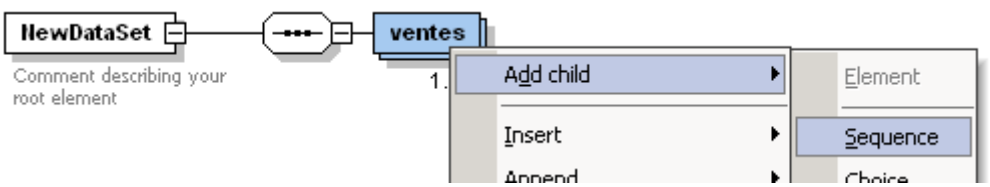
C'est très simple: à droite de la fenêtre XMLSpy vous pouvez définir cela.



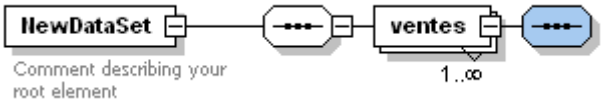
Ainsi, ne sachant pas combien il y en aura au maximum nous choisissons *unbounded* et nous définissons qu'il faut au moins une vente de le fichier XML pour que celui-ci soit considéré comme valide. Quand vous cliquez sur *unbounded*, l'élément *ventes* se change de la manière suivante:



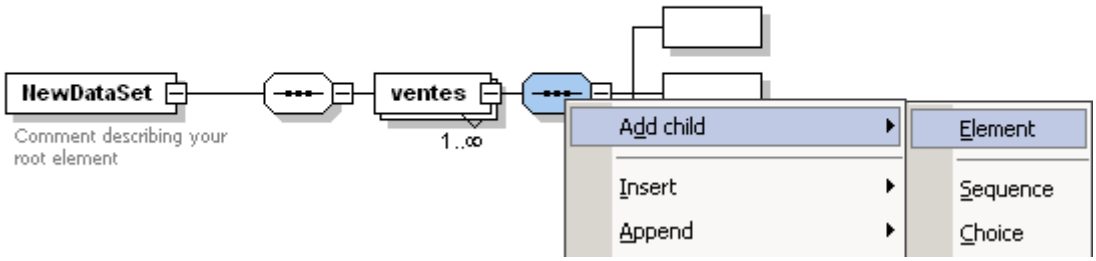
et comme *ventes* contient à nouveau une séquence de nœud nous faisons à nouveau:



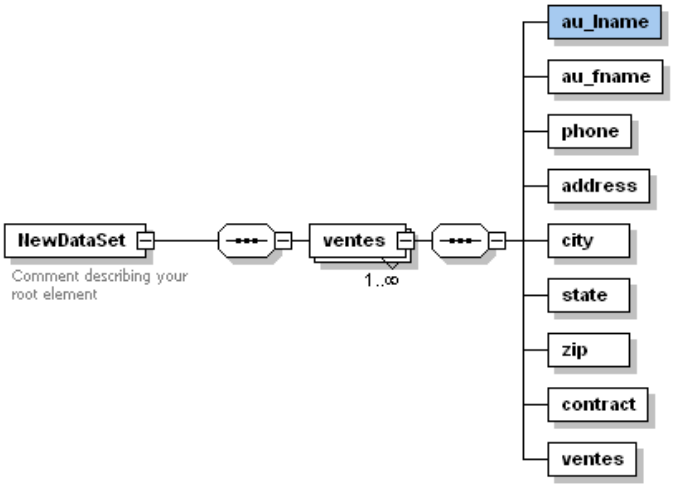
Pour obtenir:



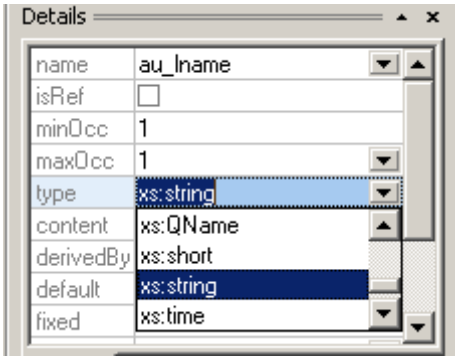
Maintenant, il nous faut ajouter les éléments enfants *au_id, au_lname, au_fname, …*. Pour ce faire, nous procédons à nouveau a peu près de la même manière que pour *ventes* (mis à part le *unbounded*) autant de fois que nécessaire:



et re-*Insert/Element* et re… et re… et re… etc. jusqu'à obtenir (après renommage des petites cases):

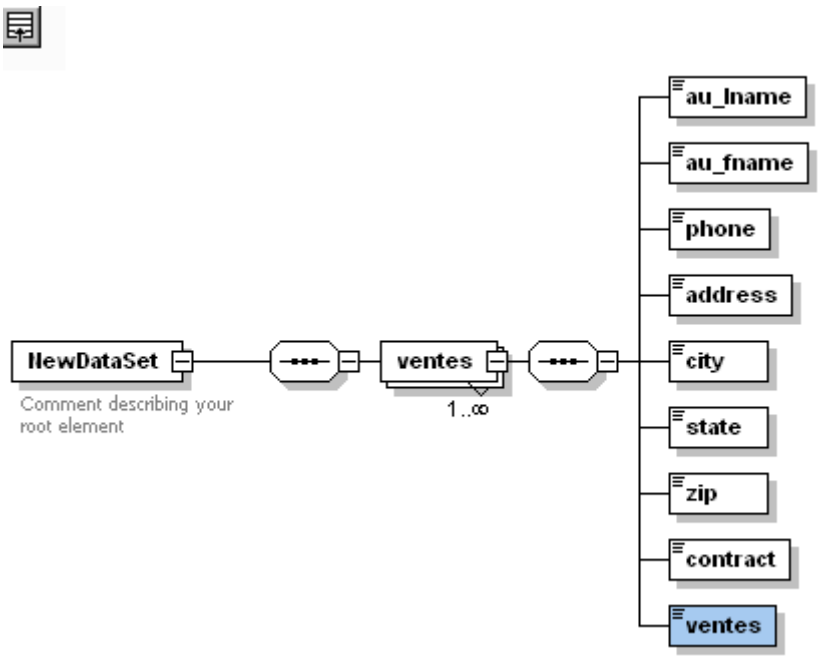


Mais ce n'est pas tout. Pour travailler vraiment proprement, il faut définit le type de données que doit/peut contenir chaque élément. Pour ce faire, à nouveau dans la partie droite de la fenêtre de XMLSpy, vous avez:



Remarque: si vous mettez le minOcc d'un élément à 0, alors celui-ci devient optionnel!

et vous pouvez choisir tous les types de données les plus important pour MS Office 2003 (dates, nombre entiers, booléens, chaîne de caractères, etc.). Une fois vos choix effectués pour chaque élément vous aurez le résultat suivant (observez bien le coin supérieur gauche des boîtes d'éléments):

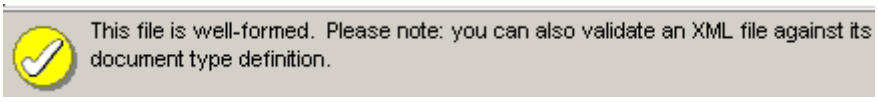


et voilà votre XSD terminé. Si vous allez maintenant dans l'ongle *Text* vous retrouverez le code tout fait tout beau. Ceci avec le moindre effort.

Et si vous liez le document XML au fichier XSD:

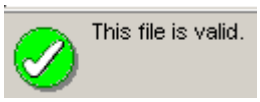
```
<?xml version="1.0" encoding="iso-8859-1"?>
<NewDataSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ventes.xsd">
```

Vous pouvez alors, toujours dans XMLSpy, vérifier la conformité (à l'aide du XSD) en cliquant sur  et vous obtiendrez:



Comme quoi il est bien conforme.

Il est aussi valide par ailleurs  (indépendamment du XSD):



Maintenant que nous sommes équipés d'un XML et du XSD nous pouvons aller dans MS Excel !

Remarques:

R1. Ce n'est pas le travail de l'employé lambda de faire le fichier XSD mais du responsable informatique (ou d'un de collègues du responsable informatique).

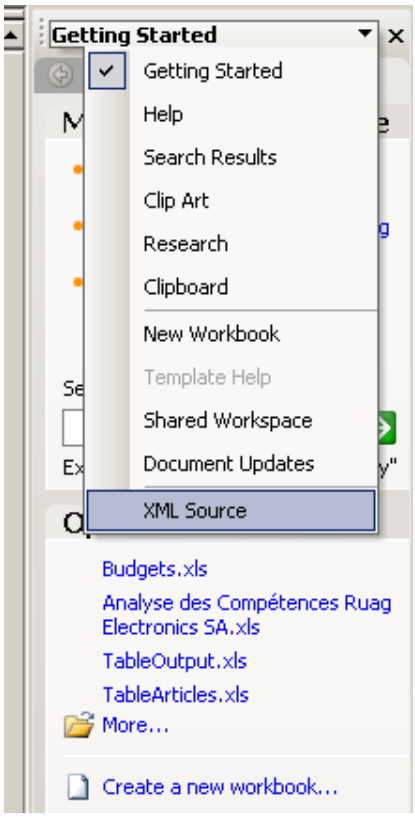
R2. Le fichier XML peut provenir de n'importe où lorsqu'il arrive dans les mains de l'employé. Le contenu du fichier XML peut même être mis à jour à distance sans que l'employé ait besoin de faire quoi que ce soit.

R3. L'employé peut très bien au besoin avec de maigres connaissances remplir à la main un fichier XML.

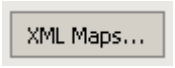
Considérons la situation suivante (qui devrait être la plus courant parmi les deux proposées): un employé reçoit un fichier XML (pour l'exemple nous prenons celui présenté tout à l'heure). Avant de travailler il faut que quelqu'un lui transmette le fichier XSD correspondant à la norme d'utilisation interne à son entreprise.

Une fois MS Excel 2003, pour importer correctement les données XML il faut procéder de la manière suivante:

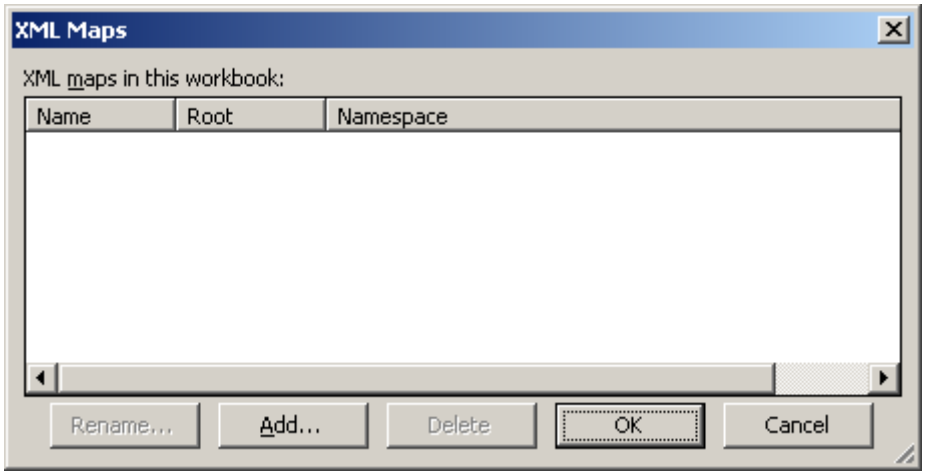
Aller dans les options du volet office (désolé j'ai pour habitude de travailler qu'avec des logiciels anglophones… il est prouvé qu'ils sont plus stables que les version en d'autres langues) et choisir l'option *Source XML (XML Source)*:



Dans le volet apparaît un bouton dans le coin inférieur droit. Cliquez dessus:



Au fait, MS Excel vous demande d'aller chercher le fichier de Mappage de votre XML correspondant au fait à votre fichier XSD. Allez donc chercher celui-ci après avoir cliqué sur le bouton *Ajouter (Add)*:



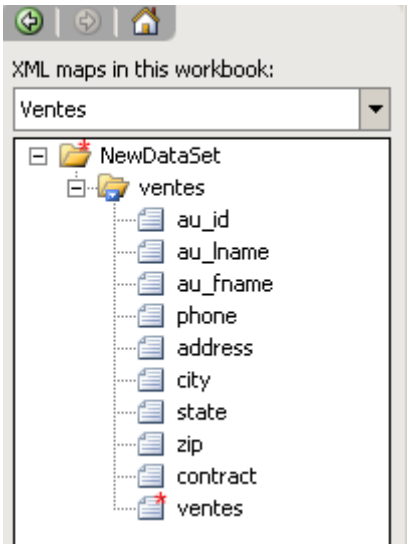
Si vous avez beaucoup de fichiers XML/XSD à traiter un conseil… renommez-les dans la fenêtre ci-dessus en cliquant sur *Renommer (Rename)*:

Cliquez ensuite sur *OK*.

Remarque: si vous voulez un namespace dans un fichier XSD (en prenant l'exemple actuel) rajoutez le texte indiqué en rouge dans la ligne suivante du fichier XSD:

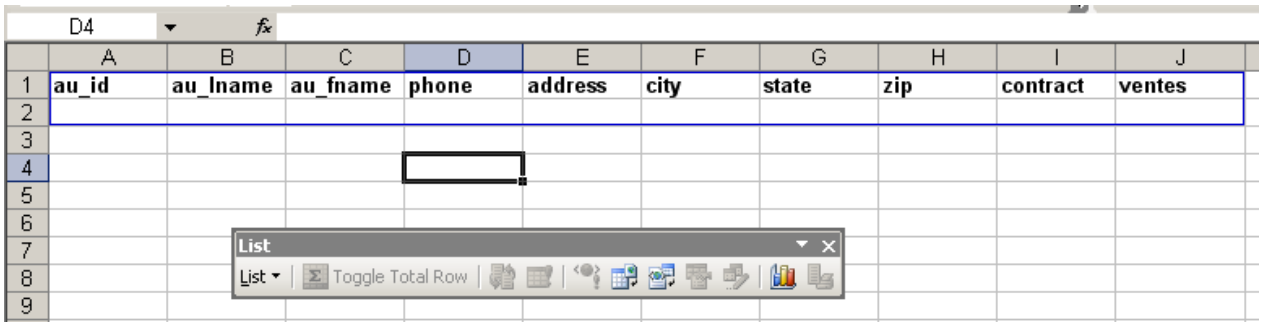
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified"
targetNamespace="http://sciences.ch">

Voilà ce qui apparaît dans le volet (le contenu dépend évidemment du schéma. ici c'est un cas extrêmement simple):



Vous remarquezer l'étoile rouge sur *Ventes*. Cela signifie simplement que dans le fichier XSD nous avons spécifié ce champ comme devant être obligatoirement informé!

Si vous souhaitez maintenant préparer l'import d'une partie ou toutes les données dans uns des feuilles de votre choix (pour les traiter ensuite comme vous le voulez avec MS Excel: filtres, tableaux croisés, consolidation, graphiques, etc.) glissez-déplacez les champs voulus dans la feuille (en prenant le petit dossier *ventes* vous prendrez tout d'un coup):



Voilà notre tableau de mappage tout prêt. Remarquez la barre d'outils spécifique aux futures données XML importées… nous y reviendrons.

Maintenant pour importer les données XML elles-mêmes, cliquez sur le bouton *Importes données XML (Import XML Data)* de votre barre d'outils *Liste (List)*:

suivez les procédure accessible à un débutant et allez chercher le fichier XML ventes.xml:

	A	B	C	D	E	F	G	H	I	J
1	au_id	au_lname	au_fname	phone	address	city	state	zip	contract	ventes
2	172-32-1176	Einstein	Albert	408 496-723	10932 Bigge Rd.	Menlo Park	CA	94025	TRUE	600
3	172-32-1233	Planck	Max	408 434-543	4304. Hollywood St.	Holl. Bvd.	CA	3454	FALSE	300
4	*									
5										

et voilà du tout cuit…

Quelques explications:

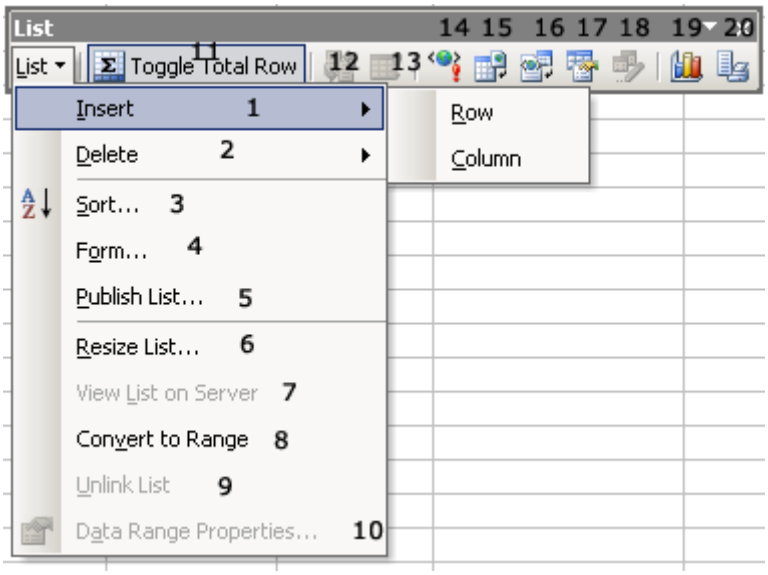
1. On peut manipuler ces données comme un tableau normal MS Excel mais tout de fois sans déplacer les données (rien n'empêche de faire des formules de liaison sur d'autres feuille ceci dit)

2. Les données sont liées au fichier MS Excel dans le sens fichier XML → Excel et non dans le sens inverse (nous verrons comment faire pour l'avoir dans l'autre sens).

   Habituellement les données dans le fichier XML sont mise à jour par un responsable mais si l'utilisateur du fichier souhait mettre alors à jour son tableau XML il n'a qu'à cliquer sur le bouton  de la barre d'outils *List* (comme pour tout import habituel dans MS Excel).

3. On peut exporter/importer ces listes dans SharePoint Portal Server.

Voyons maintenant un par un les boutons de la barre d'outils *List*:



Légendes:

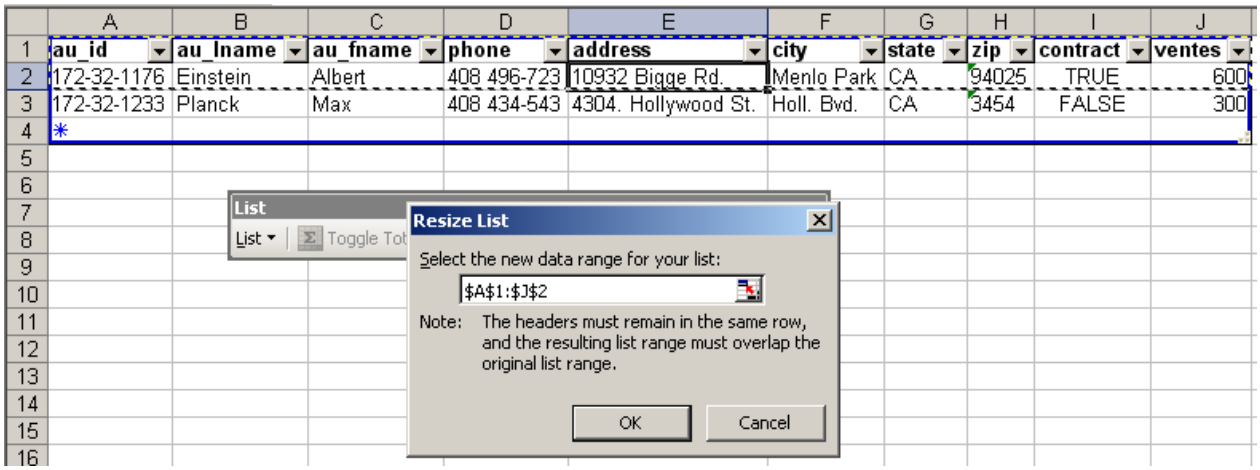
L1. & L2. Rien à dire de spécial. Ajouter/Supprime des colonnes dans la feuilles Excel mais ne modifie pas le fichier XML derrière

L3 & L4. Rien à dire. Fonction connue par tous (existe depuis 1997 dans MS Excel…)

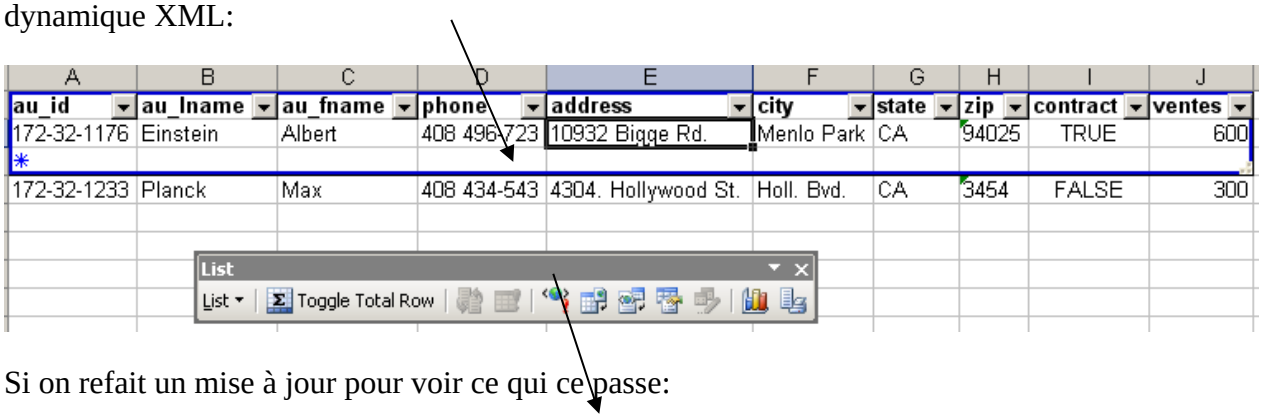
L5. Fonction valable uniquement pour les entreprises possédant Share Point Portal Server (eh oui ! désolé….)

L6. Permet de délier les zones non sélectionnées de la table du fichier XML (si vous sélectionnez les lignes 1 & 2, la ligne numéro 3 sera déliée du fichier de la zone dynamique XML). Démo:

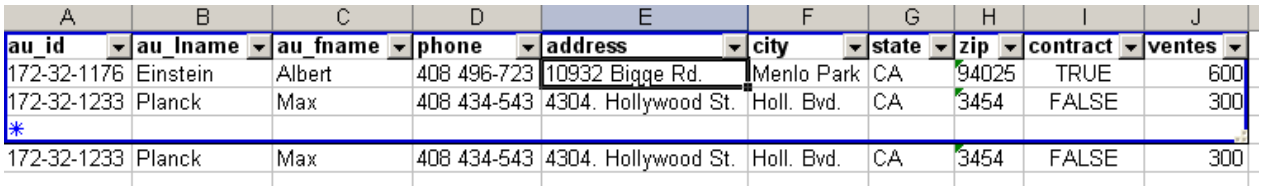
On ne sélectionne que les deux premières:



Quand on clique sur *OK*, toutes les lignes qui sont en-dessous sont "ejectées" du tableau dynamique XML:



Si on refait un mise à jour pour voir ce qui ce passe:



C.Q.F.D.

Remarque: dans votre sélection, vous devez toujours avoir la première ligne (ligne de titre).

L7. Fonction valable à nouveau uniquement pour les entreprises possédant Share Point Portal Server (désolé….)

L8. Convertit tout le tableau en zone déliée du fichier XML (c'est tout… c'est l'équivalent de L6 mais pour tout le tableau quoi…)

L9 & L10. Fonctions valables à nouveau uniquement pour les entreprises possédant Share Point Portal Server (désolé….)

L11. Pas vraiment utile… insère une somme automatique de types de données numériques (relativement à ce que vous aviez défini lors de la création du fichier XSD) dynamique qui se déplace dans le tableau au fur à mesure que des données apparaissent ou disparaissent:

au_id	au_lname	au_fname	phone	address	city	state	zip	contract	ventes
172-32-1176	Einstein	Albert	408 496-723	10932 Bigge Rd.	Menlo Park	CA	94025	TRUE	600
172-32-1233	Planck	Max	408 434-543	4304. Hollywood St.	Holl. Bvd.	CA	3454	FALSE	300
Total									**900**

L12 & L13. Fonctions valables à nouveau uniquement pour les entreprises possédant Share Point Portal Server (désolé….)

L14. On connaît déjà… Met à jour les données entre le fichier XML et Excel. Pour essayer, allez dans le fichier XML et rajoutez "à la main une ligne" et faites une mise à jour. Exemple (bof…):

au_id	au_lname	au_fname	phone	address	city	state	zip	contract	ventes
172-32-1176	Einstein	Albert	408 496-723	10932 Bigge Rd.	Menlo Park	CA	94025	TRUE	600
172-32-1233	Planck	Max	408 434-543	4304. Hollywood St.	Holl. Bvd.	CA	3454	FALSE	300
156-78-146-116	Isoz	Vincent	076 329 35 88	Ch. de Chandieu 20	Lausanne	Suisse	1006	TRUE	1200
*									
Total									**2100**

L15. On connaît déjà. Mais cependant une remarque s'impose: si vous êtes un peu "procédurier" (pardon mais y'a pas d'autres termes) vous pouvez importer des données non correspondant au fichier XSD mais alors attention pour la suite…

L16. C'est ici que notre fichier XSD joue son rôle. On peut exporter les données insérées dans un autre fichier XML et pour que celui-ci soit vraiment "propre et clean" il faut avoir importé d'abord son XSD en mappage.

Exemple: changeons (ou insérons) à la main les données du tableau ci-dessus (nous avons supprimé *Planck* et changé *Vincent* en *Aline*):

au_id	au_lname	au_fname	phone	address	city	state	zip	contract	ventes
172-32-1176	Einstein	Albert	408 496-723	10932 Bigge Rd.	Menlo Park	CA	94025	TRUE	600
156-78-146-116	Isoz	Aline	076 329 35 88	Ch. de Chandieu 20	Lausanne	Suisse	1006	TRUE	1200

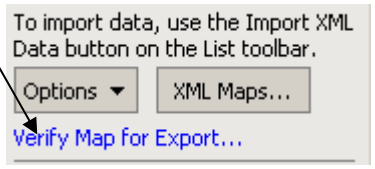
Voici le fichier XML une fois exporté:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NewDataSet>
    <ventes>
        <au_id>172-32-1176</au_id>
        <au_lname>Einstein</au_lname>
        <au_fname>Albert</au_fname>
        <phone>408 496-723</phone>
        <address>10932 Bigge Rd.</address>
        <city>Menlo Park</city>
        <state>CA</state>
        <zip>94025</zip>
        <contract>true</contract>
        <ventes>600</ventes>
    </ventes>
    <ventes>
        <au_id>156-78-146-116</au_id>
        <au_lname>Isoz</au_lname>
        <au_fname>Aline</au_fname>
        <phone>076 329 35 88</phone>
        <address>Ch. de Chandieu 20</address>
        <city>Lausanne</city>
        <state>Suisse</state>
```

```
        <zip>1006</zip>
        <contract>true</contract>
        <ventes>1200</ventes>
    </ventes>
</NewDataSet>
```

Tout est parfait !!!!

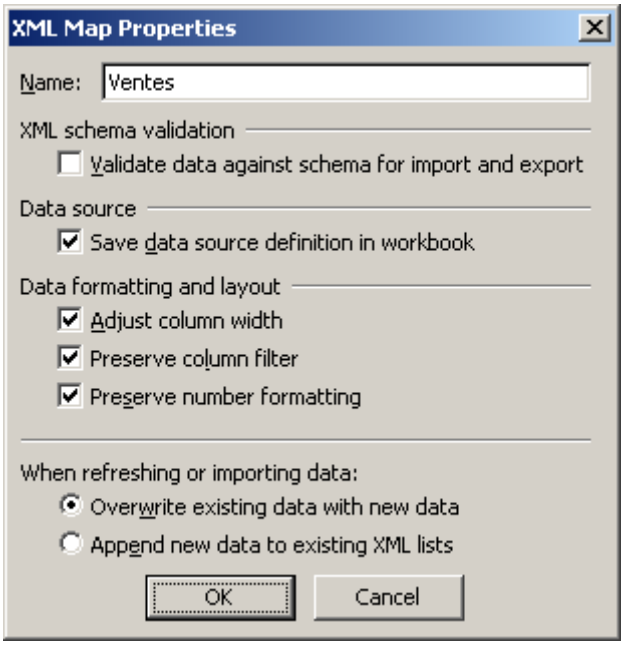
On peut d'ailleurs vérifier si le schéma est compatible pour l'export en cliquant dans le volet Office sur le lien suivant:



Dès lors si vous avez un XSD (à nouveau je rappelle que cela marche aussi sans XSD mais alors c'est pas du travail 100% pro.):

Ouf ! quand même…

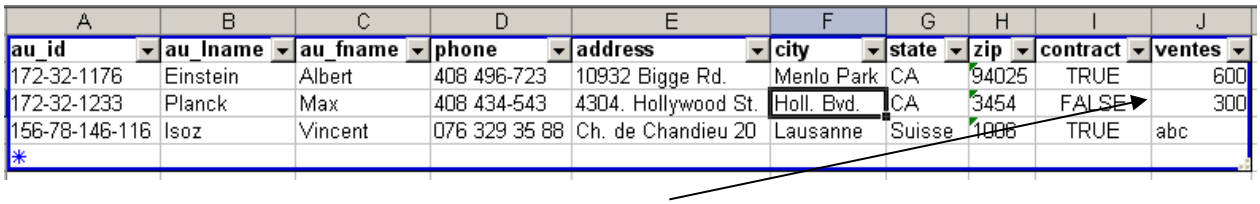
L17. Affiche les propriétés du Mapping:



Pour comprendre ce que font ces options, il suffit de lire… la seule qui peut poser problème de compréhension est la première mais c'est elle aussi qui met en avance la puissance du XSD.

Démo:

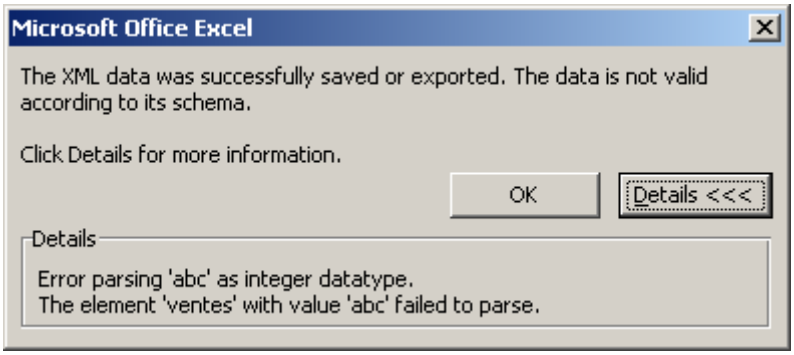
Rappelez-vous que la balise *Ventes* est du type *Integer* (nous l'avons défini ainsi dans XMLSpy). Si la première case à cocher n'est pas activée dès lors quand nous exportons le tableau suivant:



A	B	C	D	E	F	G	H	I	J
au_id	au_lname	au_fname	phone	address	city	state	zip	contract	ventes
172-32-1176	Einstein	Albert	408 496-723	10932 Bigge Rd.	Menlo Park	CA	94025	TRUE	600
172-32-1233	Planck	Max	408 434-543	4304. Hollywood St.	Holl. Bvd.	CA	3454	FALSE	300
156-78-146-116	Isoz	Vincent	076 329 35 88	Ch. de Chandieu 20	Lausanne	Suisse	1008	TRUE	abc

Cela passe comme une lettre à la poste alors que c'est…. pas bien pas bien du tout…

Notre utilisateur peut mettre des données erronées et exporter sans problèmes ça c'est pas pro! Mais si on a un bon fichier XSD et qu'on active la case à cocher *Validate data against*…. et qu'on essaie un export:



C'est pas beau ça hein !!! Voilà à quoi il sert le XSD dans MS Excel… à bien faire les choses à l'import ET à surtout à l'export.

L18. Fonction valable à nouveau uniquement pour les entreprises possédant Share Point Portal Server (décidemment… lol)

L19. & L20. Bof rien à dire… déjà connu par tout le monde

Il est bien évident que les outils présentés ci-dessus ont pour objectif d'éliminer le vieux et pitoyable et catastrophique et impropre *Enregistrer-sous* au format *XML* de MS Office 2002.

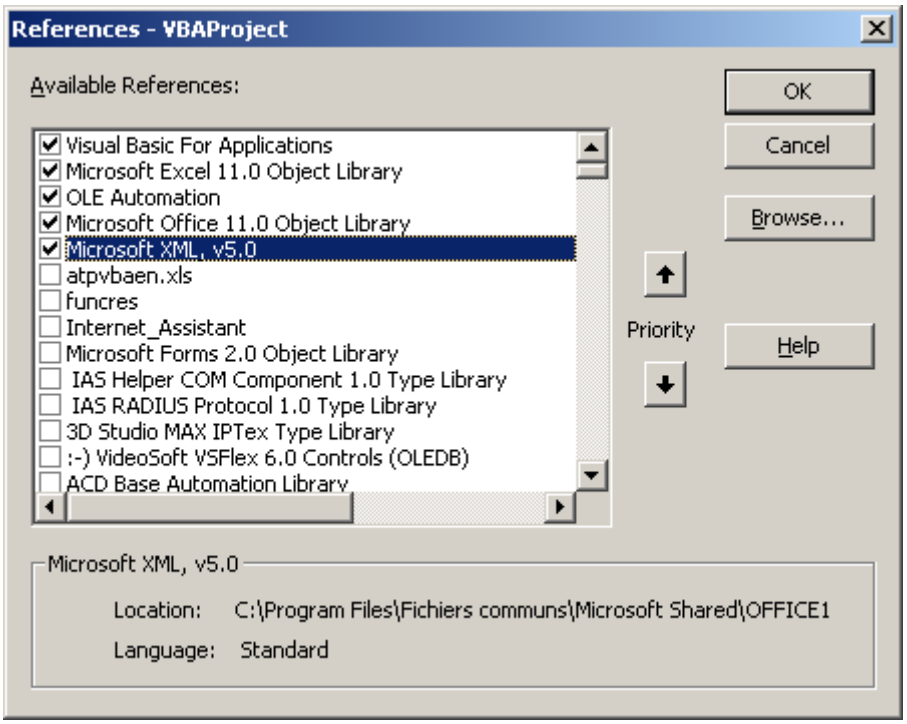
10.1.1 MS Excel 2003 VBA

Il est possible de parser un fichier XML en VBA. Cet exemple de code peut s'appliquer (après quelques petites adaptations) aussi bien à tous les autres logiciels de la suite MS Office et est bien évidemment beaucoup plus flexible qu'une simple macro enregistrée dans MS Excel 2003 ou supérieur (bien que ces dernières soient tout à fait fonctionnelles).

Considérons le fichier XML suivant nommé *test.xml* et enregistré au même endroit que le fichier Excel contenant notre code VBA:

```
<?xml version="1.0"?>
<SiteVisits>
    <Country CountryName="USA">
        <TotalVisits>1348</TotalVisits>
        <LatestVisit>1/4/2000</LatestVisit>
    </Country>
    <Country CountryName="UK">
        <TotalVisits>764</TotalVisits>
        <LatestVisit>1/4/2000</LatestVisit>
    </Country>
    <Country CountryName="Argentina">
        <TotalVisits>175</TotalVisits>
        <LatestVisit>1/2/2000</LatestVisit>
    </Country>
    <Country CountryName="Brazil">
        <TotalVisits>182</TotalVisits>
        <LatestVisit>1/4/2000</LatestVisit>
    </Country>
    <Country CountryName="Canada">
        <TotalVisits>688</TotalVisits>
        <LatestVisit>1/3/2000</LatestVisit>
    </Country>
    <Country CountryName="Denmark">
        <TotalVisits>204</TotalVisits>
        <LatestVisit>1/1/2000</LatestVisit>
    </Country>
    <Country CountryName="Germany">
        <TotalVisits>351</TotalVisits>
        <LatestVisit>1/4/2000</LatestVisit>
    </Country>
    <Country CountryName="Hong Kong">
        <TotalVisits>97</TotalVisits>
        <LatestVisit>12/30/1999</LatestVisit>
    </Country>
    <Country CountryName="Ireland">
        <TotalVisits>522</TotalVisits>
        <LatestVisit>1/4/2000</LatestVisit>
    </Country>
    <Country CountryName="Malaysia">
        <TotalVisits>14</TotalVisits>
        <LatestVisit>12/31/1999</LatestVisit>
    </Country>
```

Le code VBA suivant va importer les données dans la feuille Excel active (ne pas oublier d'ajouter les références XML au projet VBA!):




```
Sub XMLChsrger()
    'On déclare un objet de type document XML que l'on parcourra à l'aide de DOM
    'Data Object Model
   Dim oDoc As MSXML2.DOMDocument
    'Cette variable sera utilisée pour vérifier que l'on a bien chargé le fichier
   Dim fSuccess As Boolean
    'Cette variable sera utilisée pour se positionner au niveau du nœud racine du document
   Dim oRoot As MSXML2.IXMLDOMNode
    'Cette variable sera utilisée pour parcourir chaque noeud pays de l'élément racine
   Dim oCountry As MSXML2.IXMLDOMNode
    'Cette variable sera utilisée pour accéder à l'attribut (valeur) d'un noeud
   Dim oAttributes As MSXML2.IXMLDOMNamedNodeMap
   'Cette variable sera utilisée avec oAttributes pour lire la valeur du nœud Country
   Dim oCountryName As MSXML2.IXMLDOMNode
   'Cette variable sera utilisée pour parcourir les nœuds enfants de oCountry
   Dim oChildren As MSXML2.IXMLDOMNodeList
   'Cette variable sera utilisée pour accéder à un nœud enfant particulier
   Dim oChild As MSXML2.IXMLDOMNode
   'Cette variable sera utilisée pour écrire les valeurs des nœuds dans les cellules
   Dim intI As Integer
   'On créée une instance de oDoc dans la mémoir
    Set oDoc = New MSXML2.DOMDocument
    'On charge le fichier XML du disque sans le valider par du XSD et on attend qu'il soit fini
d'être chargé
   oDoc.async = False
   oDoc.validateOnParse = False
   'le fichier XML doit se trouver dans le même dossier que le fichier XSL dans cet exemple
   fSuccess = oDoc.Load(ActiveWorkbook.Path & "test.xml")
   If fSuccess Then\
      msgbox "Chargement du fichier réussi"
```

```
    End If
    'on écrit à partir de la ligne 2 car la premère contiendra les titres
    intI = 2
    'On crée la ligne de titre
    ActiveSheet.Cells(1, 1) = "Country"
    ActiveSheet.Cells(1, 2) = "Total Visits"
    ActiveSheet.Cells(1, 3) = "Latest Visit"
     'On se positionne sur le nœud racine
    Set oRoot = oDoc.documentElement
     ' Pour chaque nœud enfant direct de la racine…
    For Each oCountry In oRoot.childNodes
      ' On se positionne sur le noeud
      Set oAttributes = oCountry.Attributes
      'On extrait de lnom de country…
      Set oCountryName = oAttributes.getNamedItem("CountryName")
      'et on le place sur la feuille
      ActiveSheet.Cells(intI, 1).Value = oCountryName.Text
      ' pour le pays en cours on parcoure tous les nœuds enfants
      Set oChildren = oCountry.childNodes
      For Each oChild In oChildren
        ' On vérifie si le noeud est bien celui que l'on recherche pour le positionner dans la bonne
colonne
        If oChild.nodeName = "TotalVisits" Then
          ActiveSheet.Cells(intI, 2) = oChild.nodeTypedValue
        ElseIf oChild.nodeName = "LatestVisit" Then
          ActiveSheet.Cells(intI, 3) = oChild.nodeTypedValue
        End If
      'On passe au prochain élément enfant de Country
      Next oChild
      intI = intI + 1
      'On passe au prochain élément enfant de Country
    Next oCountry
End Sub
```

Le résultat obtenu sera alors le suivant:

Country	Total Visits	Latest Visit
USA	1348	04.01.2000
UK	764	04.01.2000
Argentina	175	02.01.2000
Brazil	182	04.01.2000
Canada	688	03.01.2000
Denmark	204	01.01.2000
Germany	351	04.01.2000
Hong Kong	97	30.12.1999
Ireland	522	04.01.2000
Malaysia	14	31.12.1999
Netherlands	542	04.01.2000
New Zealand	599	03.01.2000
Norway	452	03.01.2000
Scotland	538	04.01.2000
Sweden	422	02.01.2000
Wales	301	01.01.2000
Yugoslavia	37	30.12.1999
Zambia	42	04.01.2000

et pour l'export en XML on pourra utiliser la technique (très vieille car déjà utilisée à l'époque pour exporter en CSV, TXT ou HTML) suivante:

```
Sub EcrireFichierXML()

    Dim strPath As String

    'cette boîte de dialogue nous renvoie le nom du fichier saisi ainsi que le chemin !
    strPath = Application.GetSaveAsFilename(fileFilter:="XML files (*.xml), *.xml")
    MsgBox strPath
    Open strPath For Output As #1
    Print #1, "<?xml version=""1.0"" encoding=""iso-8859-1""?>"
    'on pourrait très bien rajouter ici la référence à un fichier XSL pour faire des superbes
rapports!
    Print #1, "<SiteVisits>"
    For i = 1 To Range("A1").End(xlDown).Row Step 1
        Print #1, "<Country CountryName=""" & Cells(i, 1) & """>"
        Print #1, "<TotalVisits>" & Cells(i, 2) & "</TotalVisits>"
        Print #1, "<LatestVisit>" & Cells(i, 3) & "</LatestVisit>"
        Print #1, "</Country>"
    Next i
    Print #1, "</SiteVisits>"
    Close #1

End Sub
```

10.2 MS Word 2003

L'usage de XML dans MS Word 2003 et un peu plus subtil que dans MS Excel 2003 et pour faire un excellent usage du XML il est préférable d'avoir d'excellentes connaissances du CSS ou du XSL.

Par ailleurs, avec les outils standards mis à disposition, le XML est un peu à "usage unique". On entend par là, qu'il n'y pas de nœud avec occurrences normalement pour l'usage du XML dans MS Word.

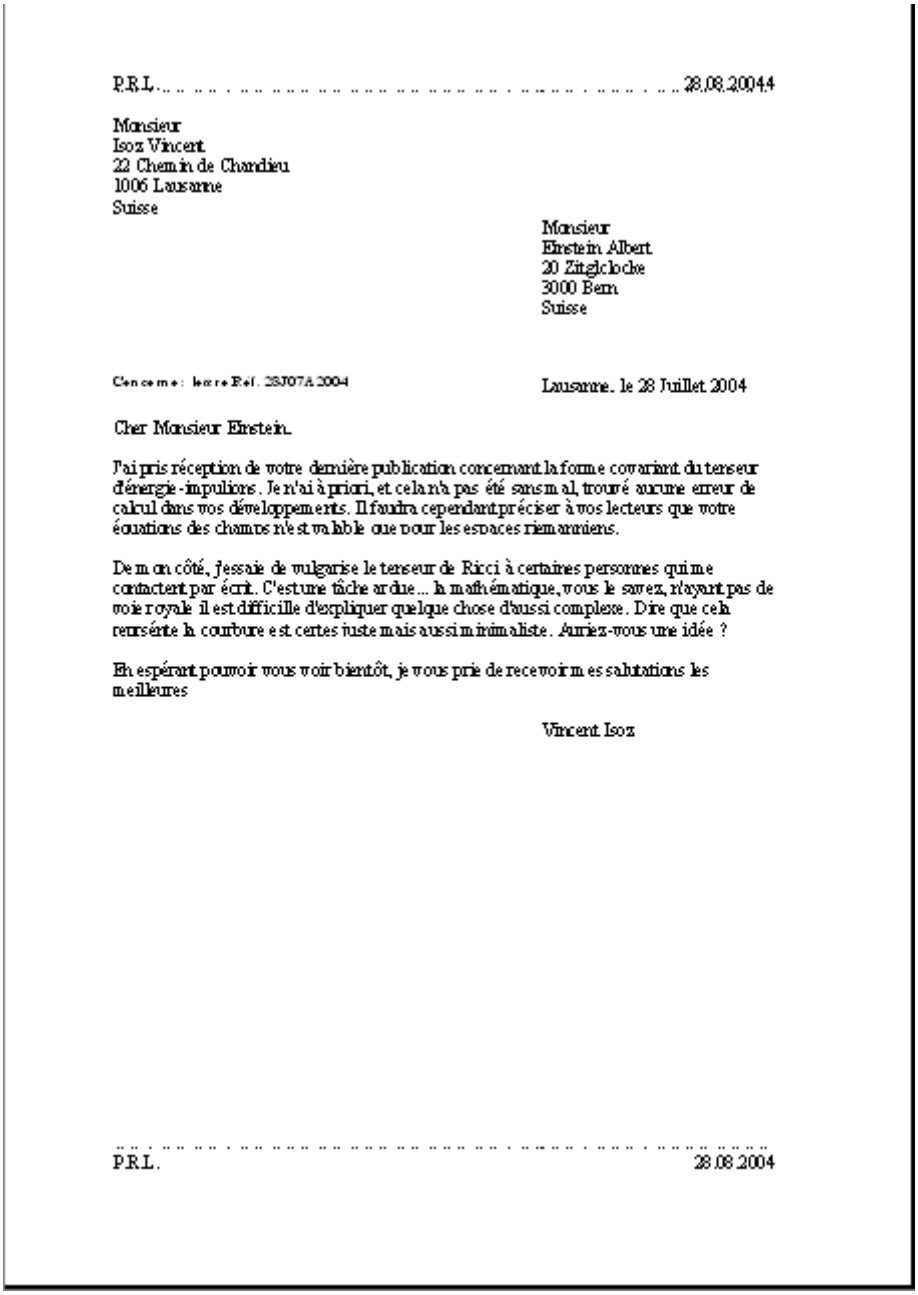
Au fait, la stratégie de Microsoft semble être de faire de MS Word une interface d'une GED (Gestion électronique de documentation): des employés saisissent des infos dans une base de données, celle-ci génère un fichier XML ultra-léger et "full compliant" qui permet de créer un document MS Word pré-formaté ou dans l'autre sens.

Remarque: il est malheureux que Microsoft n'ai pas pensé au publipostage avec le XML. C'est assez incompréhensible (il existe des outils payants qui font cela mais quand même…). Par ailleurs le connexion à des bases de données externes ne se fait pas avec l'XML (autre bizarrerie !!!)

Mais voyons quand même de quoi MS Word 2003 est capable avec deux exemples:

10.2.1 Lettre type (exemple 1)

Voici une lettre type (je sais elle est moche mais pour le présent support je voulais pas faire quelque chose de compliqué):



P.R.L., 28.08.2004

Monsieur
Isoz Vincent
22 Chemin de Chandieu
1006 Lausanne
Suisse

Monsieur
Einstein Albert
20 Zitglcloche
3000 Bern
Suisse

Concerne : lettre Réf. 28J07A2004

Lausanne, le 28 Juillet 2004

Cher Monsieur Einstein,

J'ai pris réception de votre dernière publication concernant la forme covariant du tenseur d'énergie-impulions. Je n'ai à priori, et cela n'a pas été sans mal, trouré aucune erreur de calcul dans vos développements. Il faudra cependant préciser à vos lecteurs que votre équations des champs n'est valable que pour les espaces riemanniens.

De mon côté, j'essaie de vulgarise le tenseur de Ricci à certaines personnes qui me contactent par écrit. C'est une tâche ardue... la mathématique, vous le savez, n'ayant pas de voie royale il est difficille d'expliquer quelque chose d'aussi complexe. Dire que cela représente la courbure est certes juste mais aussi minimaliste. Auriez-vous une idée ?

En espérant pouvoir vous voir bientôt, je vous prie de recevoir mes salutations les meilleures

Vincent Isoz

P.R.L. 28.08.2004

Le but de cet exemple: gérer le contenu en XML pour que les employés n'aient plus à faire la mise en page.

Le responsable de la documentation de l'entreprise crée le fichier XSL (LettreType.xsl) de mise en page suivant (relativement à l'exemple):

```
<?xml version="1.0" encoding="iso-8859-1"?>
  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head>
<title>Lettre Type</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<link href="lettretype.css" rel="stylesheet" type="text/css" />
</head>

<body>
<table width="581" border="0" cellspacing="0" cellpadding="0">
```

```
<tr>
  <td class="entete"><xsl:value-of select="lettretype/entetegauche"/></td>
  <td class="entete"><div align="right"><xsl:value-of select="lettretype/entetedroit"/>4</div></td>
</tr>
<tr>
  <td><br></br></td>
  <td><br></br></td>
</tr>
<tr>
  <td width="406"><p><xsl:value-of select="lettretype/titreexpediteur"/><br />
      <xsl:value-of select="lettretype/nomprenomexpediteur"/><br />
      <xsl:value-of select="lettretype/adressexpediteur"/><br />
      <xsl:value-of select="lettretype/cpvilleexpedtieur"/><br />
      <xsl:value-of select="lettretype/paysexpediteur"/></p></td>
  <td width="175"></td>
</tr>
<tr>
  <td></td>
  <td><xsl:value-of select="lettretype/titredestinataire"/><br />
    <xsl:value-of select="lettretype/nomprenomdestinataire"/><br />
    <xsl:value-of select="lettretype/adressedestinataire"/><br />
    <xsl:value-of select="lettretype/cpvilledestinataire"/><br />
    <xsl:value-of select="lettretype/paysdestinataire"/></td>
</tr>
<tr>
  <td height="50"></td>
  <td><br></br></td>
</tr>
<tr>
  <td height="19" class="gras"><xsl:value-of select="lettretype/concernce"/>      </td>
  <td><xsl:value-of select="lettretype/villedate"/></td>
</tr>
<tr>
  <td><br></br></td>
  <td><br></br></td>
</tr>
<tr>
  <td><xsl:value-of select="lettretype/politesse"/></td>
  <td><br></br></td>
</tr>
<tr>
  <td><br></br></td>
  <td><br></br></td>
</tr>
<tr>
  <td colspan="2"><xsl:value-of select="lettretype/paragraphe1"/></td>
</tr>
<tr>
  <td><br></br></td>
  <td><br></br></td>
</tr>
<tr>
  <td colspan="2"><xsl:value-of select="lettretype/paragraphe2"/></td>
</tr>
<tr>
  <td><br></br></td>
  <td><br></br></td>
</tr>
<tr>
  <td colspan="2"><xsl:value-of select="lettretype/salutations"/></td>
</tr>
<tr>
  <td><br></br></td>
  <td><br></br></td>
</tr>
<tr>
  <td><br></br></td>
  <td><xsl:value-of select="lettretype/signature"/></td>
</tr>
<tr>
  <td><br></br></td>
```

```
    <td><br></br></td>
  </tr>
  <tr>
    <td height="341"></td>
    <td><br></br></td>
  </tr>
  <tr>
    <td height="19" valign="bottom" class="pieddepage"><xsl:value-of select="lettretype/piedgauche"/></td>
    <td valign="bottom" class="pieddepage"><div align="right"><xsl:value-of select="lettretype/pieddroit"/></div></td>
  </tr>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Dont voici le contenu du petit fichier CSS (lettretype.css) correspondant (il met les bordures des en-têtes et pieds de page, et le *Concerne* en gras):

```
.entete {
    border-top-style: none;
    border-right-style: none;
    border-bottom-style: dotted;
    border-left-style: none;
    border-bottom-width: thin;
    border-bottom-color: #000000;
}
.pieddepage {
    border-top-width: thin;
    border-right-width: thin;
    border-bottom-width: thin;
    border-left-width: thin;
    border-top-style: dotted;
    border-top-color: #000000;
    border-right-color: #000000;
    border-bottom-color: #000000;
    border-left-color: #000000;
}
.gras {
    font-family: "Times New Roman", Times, serif;
    font-weight: bold;
    font-size: 14px;
}
```

et enfin le fichier XML (lettretype.xml) correspondant:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="LettreType.xsl"?>
<lettretype>

<entetegauche>P.R.L.</entetegauche>
<entetedroit>28.08.2004</entetedroit>

<titreexpediteur>Monsieur</titreexpediteur>
<nomprenomexpediteur>Isoz Vincent</nomprenomexpediteur>
<adressexpediteur>22 Chemin de Chandieu</adressexpediteur>
<cpvilleexpedtieur>1006 Lausanne</cpvilleexpedtieur>
<paysexpediteur>Suisse</paysexpediteur>

<titredestinataire>Monsieur</titredestinataire>
```

```
<nomprenomdestinataire>Einstein Albert</nomprenomdestinataire>
<adressedestinataire>20 Zitglclocke</adressedestinataire>
<cpvilledestinataire>3000 Bern</cpvilledestinataire>
<paysdestinataire>Suisse</paysdestinataire>

<concernce>Concerne: lettre Réf. 28J07A2004 </concernce>
<villedate>Lausanne, le 28 Juillet 2004</villedate>

<politesse>Cher Monsieur Einstein,</politesse>

<paragraphe1>J'ai pris réception de votre dernière publication concernant la forme covariant du tenseur d'énergie-impulions. Je n'ai à priori, et cela n'a pas été sans mal, trouvé aucune erreur de calcul dans vos développements. Il faudra cependant préciser à vos lecteurs que votre équations des champs n'est valable que pour les espaces riemanniens.</paragraphe1>

<paragraphe2>De mon côté, j'essaie de vulgarise le tenseur de Ricci à certaines personnes qui me contactent par écrit. C'est une tâche ardue... la mathématique, vous le savez, n'ayant pas de voie royale il est difficille d'expliquer quelque chose d'aussi complexe. Dire que cela reprsénte la courbure est certes juste mais aussi minimaliste. Auriez-vous une idée ?</paragraphe2>

<salutations>En espérant pouvoir vous voir bientôt, je vous prie de recevoir mes salutations les meilleures</salutations>

<signature>Vincent ISOZ</signature>

<piedgauche>P.R.L.</piedgauche>
<pieddroit>28.08.2004</pieddroit>

</lettretype>
```

et cela donne exactement la page (réf. capture d'écran de la lettre type) que vous avez vu tout à l'heure.

Avantages jusqu'à maintenant (pour les documents types):

1. L'employé n'a pas besoin de connaître les techniques de mise en page pour créer une lettre propre. Il lui suffit d'ouvrir le fichier XML et de saisir les infos entre les balises (exercice dont un enfant de 7 ans serait capable d'après le consortium du W3C).

2. Les documents gérés sont très légers. Effectivement: les 3 fichers XML + XSL + CSS font 5.27 Ko à eux trois alors que le fichier MS Word correspondant fait environ 30 Ko soit 6 fois plus !!!!!!!!! (imaginez une PME qui fait environ 50'000 à 60'000 documents par année le gain de place et de productivité !!!)

3. Les données XML peuvent êtres centralisées dans n'importe quel système de Base de données ou être ré-utilisables dans n'importe quel logiciel compatible avec XML.

4. Les données XML satisfont bien évidemment les protocoles de sécurités pour les flux XML des webservices (au besoin…)

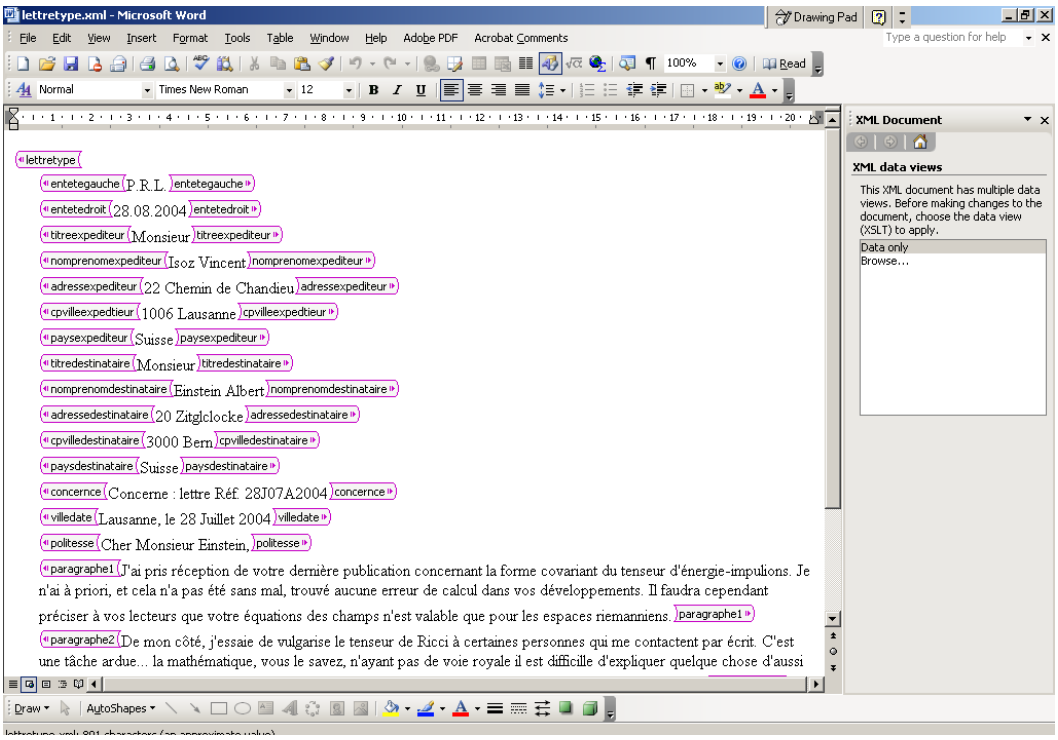
5. On peut appliquer à tout moment une nouvelle mise en page différente de la première en créant un nouveau fichier XSL sans avoir besoin de ressaisir les données.

Bon et puis MS Word dans tout cela ? Nous y venons. Avant, enlevez de votre fichier XML la ligne suivante

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

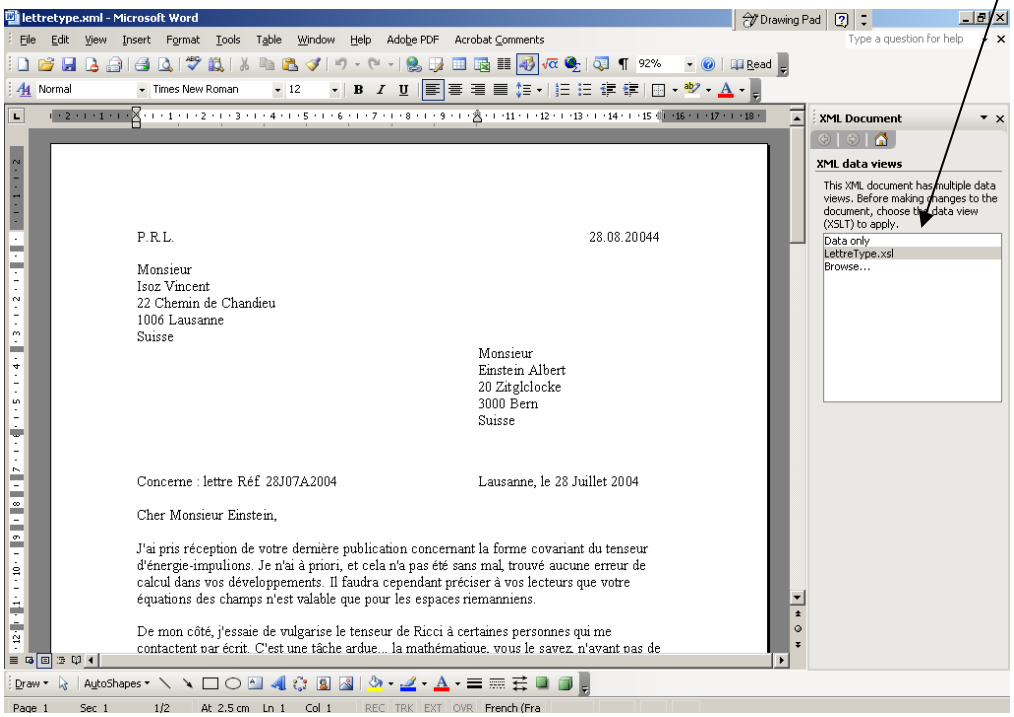
Maintenant, ouvrons notre fichier XML dans MS Word 2003 en allant dans *Fichier/Ouvrir* chercher lettretype.xml.

Vous avez donc la possibilité de changer le contenu du fichier XML depuis MS Word (mais ce n'est pas obligatoire… si vous voyez où je veux en en venir…):



On voit ci-dessus en rose les balises XML qui respectent d'ailleurs par leur position les hiérarchies du fichier XML. On voit également à droite de l'écran une option *Parcourir (Browse)* et *Données seulement (Data only)*. Eh bien… cliquons sur *Browse* et allons chercher notre document XSL (c'est ce que fera l'employé normalement).

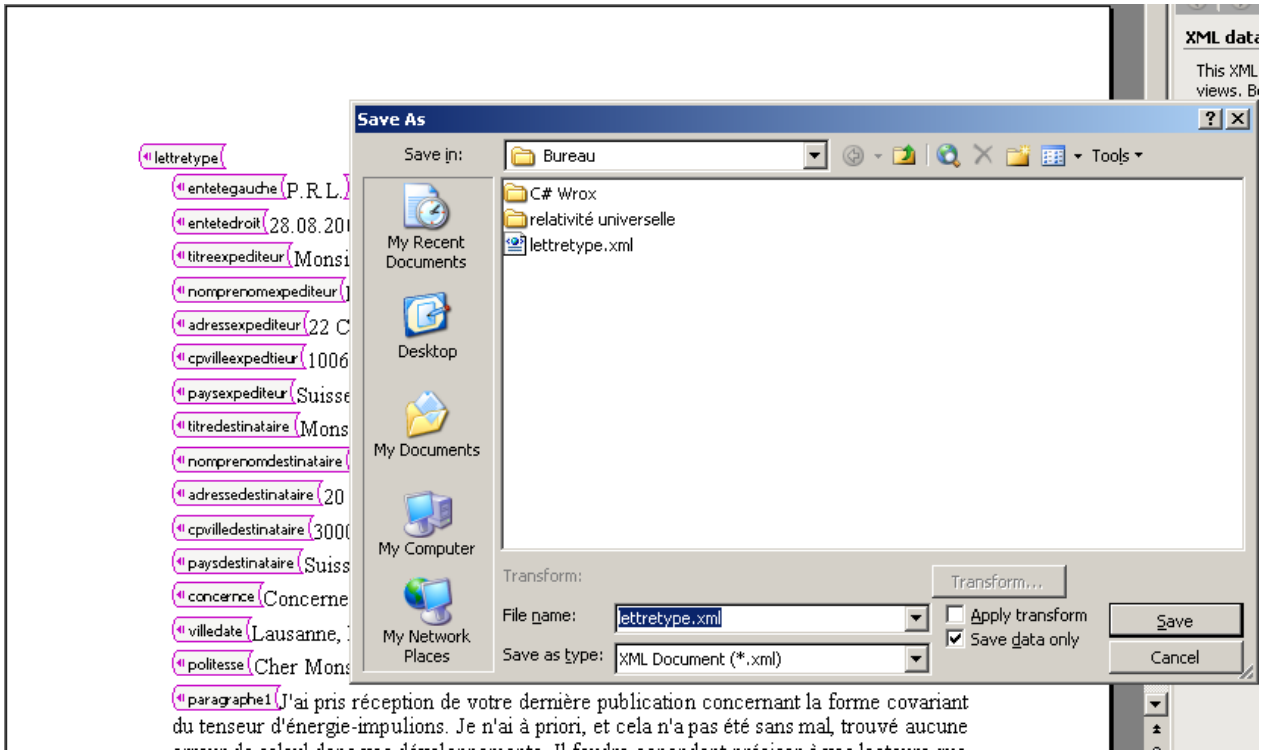
Le résultat dans MS Word est alors (c'est pas puissant ça !!!????):



et vous pouvez à tout moment rebasculer en mode *Data Only* pour changer les données (cela m'émeut tellement c'est beau…) ou appliquer un autre fichier XSL.

Le responsable de la documentation peut changer aussi à tout moment le fichier CSS pour que cela s'applique à la centaine voir millier de documents de la centaine ou millier d'employés de son entreprise (imaginez !!!).

Si l'employé change les données du fichier XML d'origine dans MS Word, il peut à tout moment aller dans *Enregistrer sous* et voici ce qui apparaît:



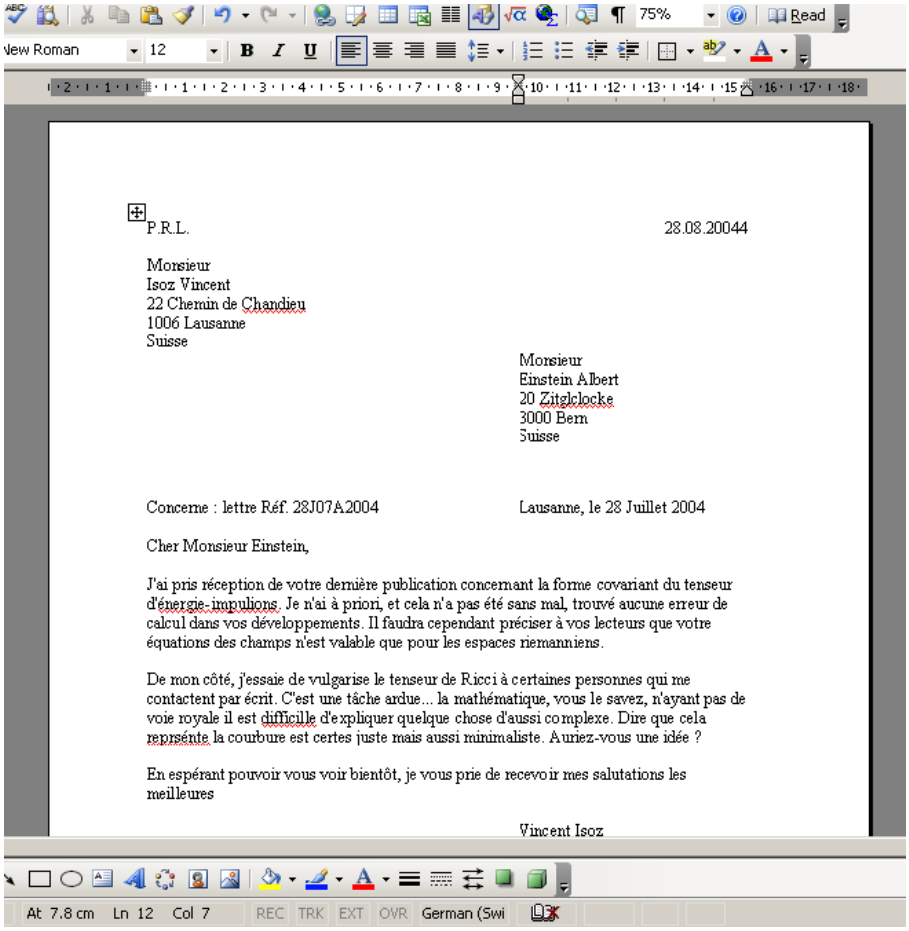
Donc il peut par défaut (car l'option *Save Data Only* est activée) enregistrer les nouvelles données dans un nouveau fichier XML (qui sera parfaitement propre contrairement à la m… que faisait MS Office 2002) pour le renvoyer au responsable informatique pour que celles-ci soient aussi centralisées dans une GED pour l'envoyer à une personne qui ne pourrait pas ouvrir les fichiers *.doc, *.rtf, etc..

Attardons-nous sur l'option: *Apply Transform*

Pour l'utiliser il faut d'abord:

1. Être en mode *Data Only*
2. Aller dans *Enregistrer sous* et cocher les deux cases *Save Data Only* ET *Apply Transform*
3. Cliquer sur le bouton *Transform* pour aller cherche le fichier *XSL*
4. choisir un nom de fichier… et enregistrer
5. Changer l'extension du fichier résultant de .xml à .htm

Résultat ? Un fichier à l'extension *.xml mais ne contenant que du HTML ! Donc un document simple préformaté avec les données et lorsqu'on ouvre ce fichier dans MS Word (*Fichier/Ouvrir*):



P.R.L. 28.08.20044

Monsieur
Isoz Vincent
22 Chemin de Chandieu
1006 Lausanne
Suisse

Monsieur
Einstein Albert
20 Zitglclocke
3000 Bern
Suisse

Concerne : lettre Réf. 28J07A2004 Lausanne, le 28 Juillet 2004

Cher Monsieur Einstein,

J'ai pris réception de votre dernière publication concernant la forme covariant du tenseur d'énergie-impulions. Je n'ai à priori, et cela n'a pas été sans mal, trouvé aucune erreur de calcul dans vos développements. Il faudra cependant préciser à vos lecteurs que votre équations des champs n'est valable que pour les espaces riemanniens.

De mon côté, j'essaie de vulgarise le tenseur de Ricci à certaines personnes qui me contactent par écrit. C'est une tâche ardue... la mathématique, vous le savez, n'ayant pas de voie royale il est difficille d'expliquer quelque chose d'aussi complexe. Dire que cela représénte la courbure est certes juste mais aussi minimaliste. Auriez-vous une idée ?

En espérant pouvoir vous voir bientôt, je vous prie de recevoir mes salutations les meilleures

Vincent Isoz

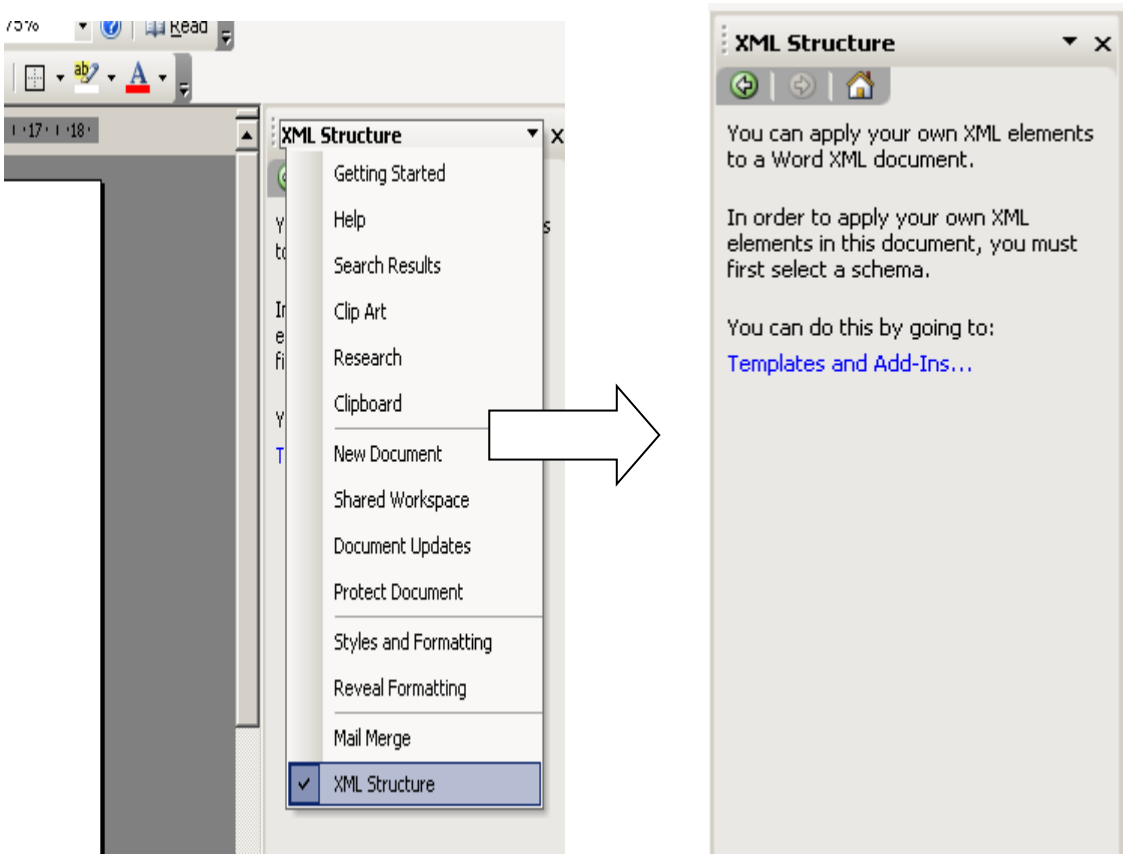
et voilà ! Il suffit d'envoyer cela à un collègue qui ne sait pas comment appliquer dans MS Word des fichiers XSL sur des XML pour qu'il ait la forme finale du document.

Remarque: au peut au besoin aussi changer l'extension de *.xml en *.htm pour avoir une simple page web publiable sur n'importe que intra ou extranet.

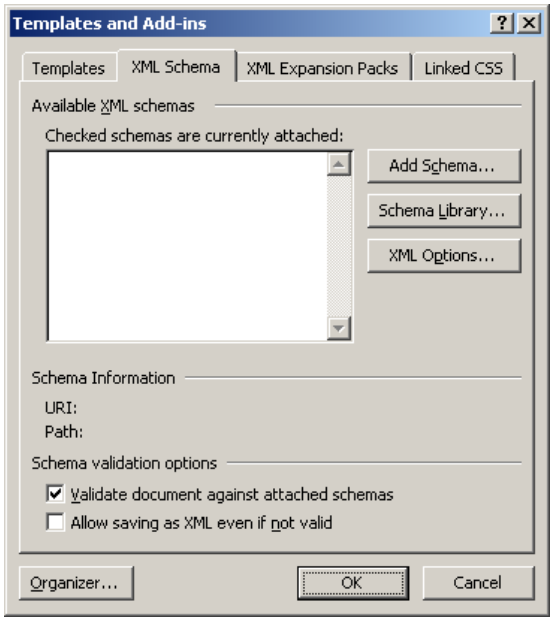
Voilà ! C'était le premier exemple. Passons au deuxième.

10.2.2 Lettre type (exemple 2)

On ouvre MS Word et on va dans l'onglet Office choisir *Structure XML (XML Structure)*:



Quand vous cliquez sur *Templates and Add-ins* voici ce qui apparaît:



Les onglets *Modèles (Templates)*, *CSS Liés (Linked CSS)*, existent déjà depuis la version 2002 de MS Office 2000 et ne concernent pas vraiment le XML (bien que… mais cela dépasse le cadre de ce support).

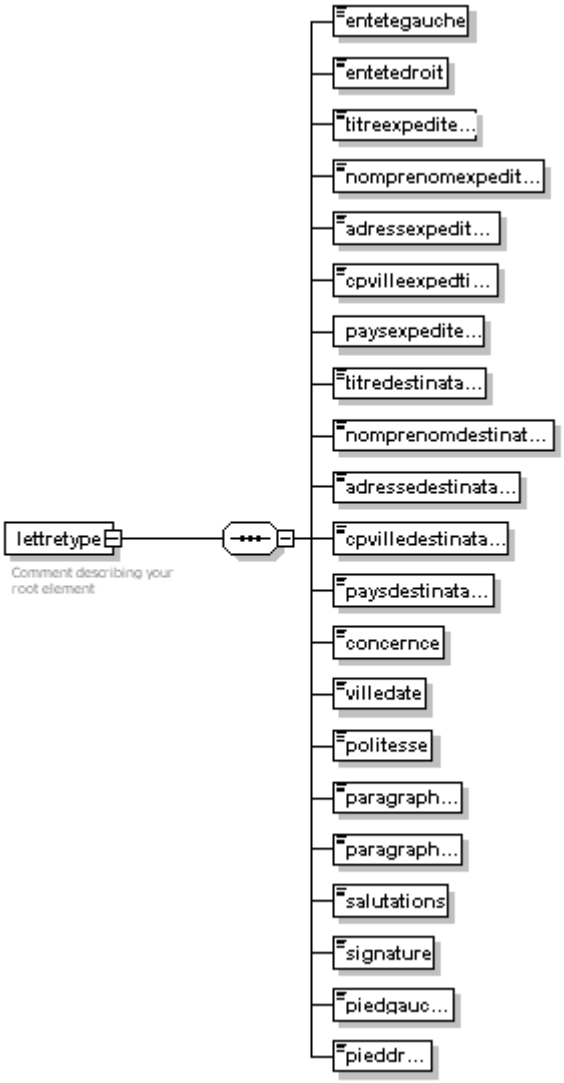
Le troisième onglet *XML Extension Packs* sort également du contexte de ce support. Il concerne les développeurs .Net qui créent des documents d'un nouveau type sous MS Office 2003 que nous appelons des "Smart-Documents". C'est beaucoup plus complexe et beaucoup

plus puissant que tout ce que nous avons vu jusqu'à maintenant donc nous ne nous y attarderons pas non plus.

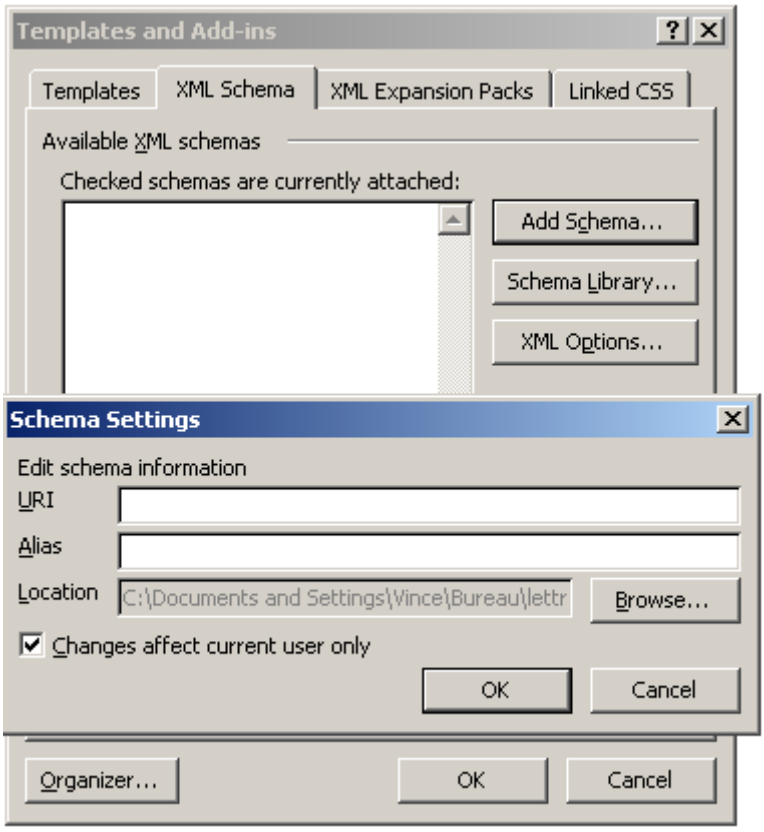
Donc finalement (et c'est déjà pas… mal), nous nous limiterons à l'étude de l'onglet *Schéma XML (XML Schema)*.

Qu'est-ce qu'un schéma… et ben… rigoureusement… (parce qu'on peut le faire de manière dégueulasse aussi…) c'est un XSD ! Donc créons de suite le fichier XSD de notre lettre type dans XMLSpy:

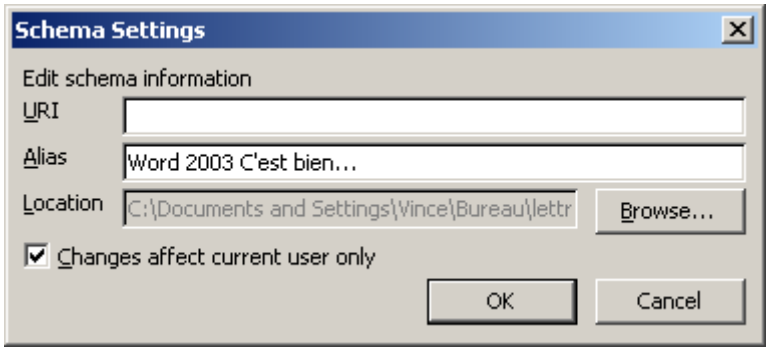
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="lettretype">
        <xs:annotation>
            <xs:documentation>Comment describing your root element</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="entetegauche" type="xs:string"/>
                <xs:element name="entetedroit" type="xs:string"/>
                <xs:element name="titreexpediteur" type="xs:string"/>
                <xs:element name="nomprenomexpediteur" type="xs:string"/>
                <xs:element name="adressexpediteur" type="xs:string"/>
                <xs:element name="cpvilleexpedtieur" type="xs:string"/>
                <xs:element name="paysexpediteur"/>
                <xs:element name="titredestinataire" type="xs:string"/>
                <xs:element name="nomprenomdestinataire" type="xs:string"/>
                <xs:element name="adressedestinataire" type="xs:string"/>
                <xs:element name="cpvilledestinataire" type="xs:string"/>
                <xs:element name="paysdestinataire" type="xs:string"/>
                <xs:element name="concernce" type="xs:string"/>
                <xs:element name="villedate" type="xs:string"/>
                <xs:element name="politesse" type="xs:string"/>
                <xs:element name="paragraphe1" type="xs:string"/>
                <xs:element name="paragraphe2" type="xs:string"/>
                <xs:element name="salutations" type="xs:string"/>
                <xs:element name="signature" type="xs:string"/>
                <xs:element name="piedgauche" type="xs:string"/>
                <xs:element name="pieddroit" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```



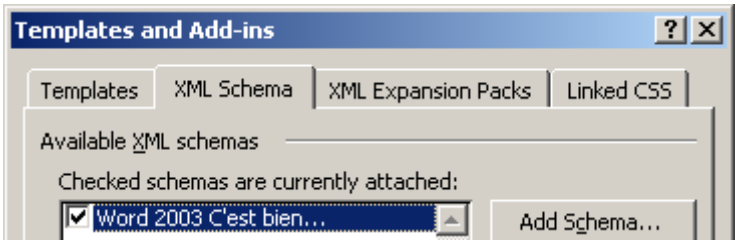
et maintenant revenons à MS Word 2003. Cliquez sur le bouton *Add Schema* de la fenêtre *Templates and Add-ins*. Une fois que vous avez cliqué sur le fichier LettreType.xsd la fenêtre suivante apparaît:



Dans l'URI faites un espace vide (barre d'espacement du clavier) et dans l'Alias mettez un joli nom pour votre fichier XSD:

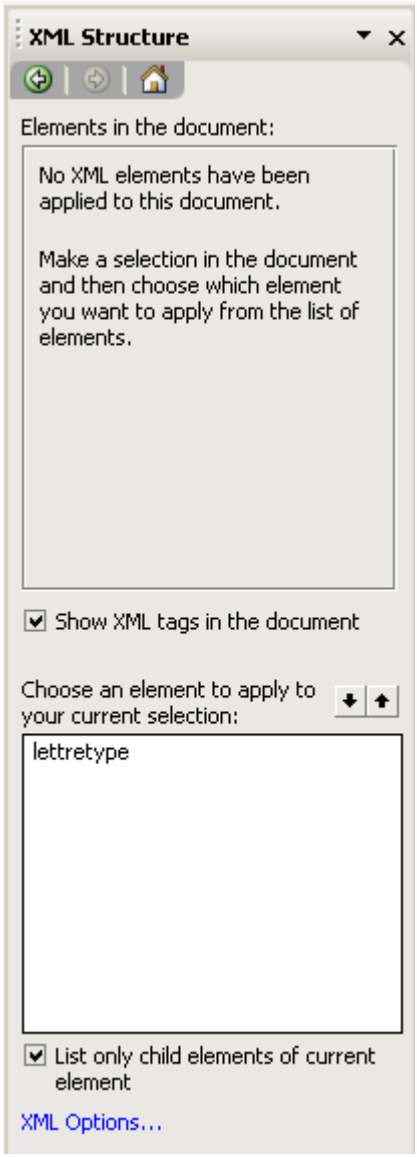


Quand vous cliquez sur *OK* vous aurez:



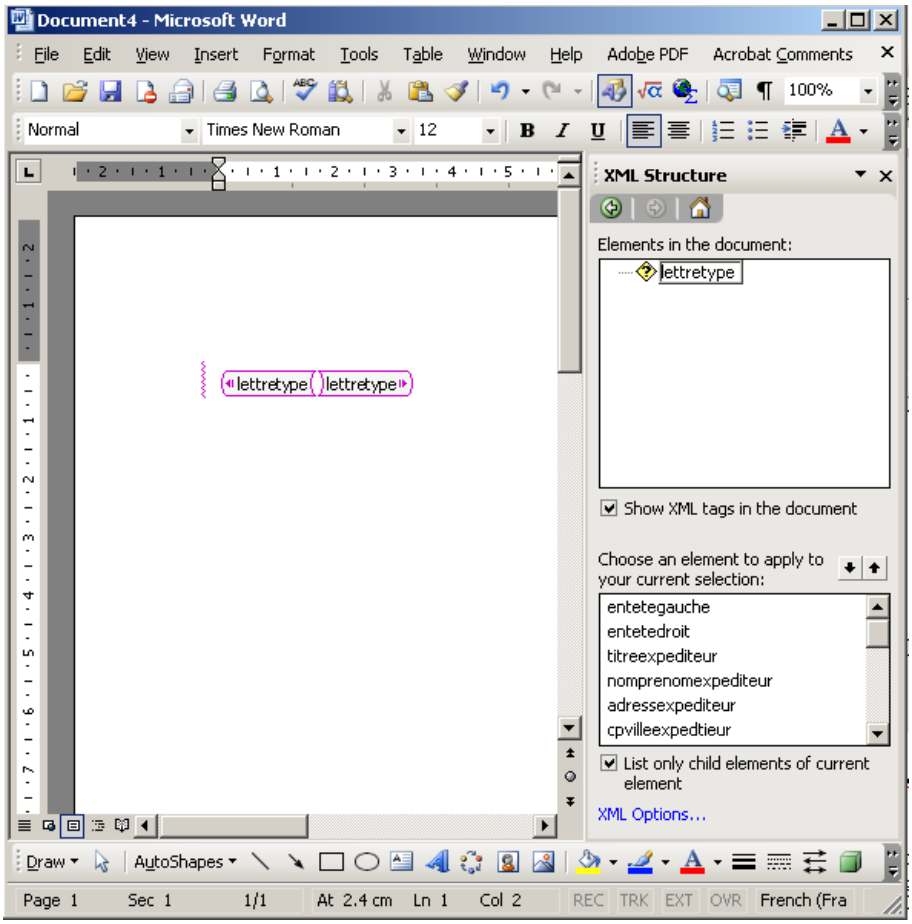
Donc comme vous pouvez le voir, MS Word 2003 vous propose un petit gestionnaire de fichier XSD et quand vous souhaitez en utiliser un, il suffit de cocher la petite case correspondante.

Quand vous cliquez sur *OK*, voici ce qui apparaît dans le volet Office:

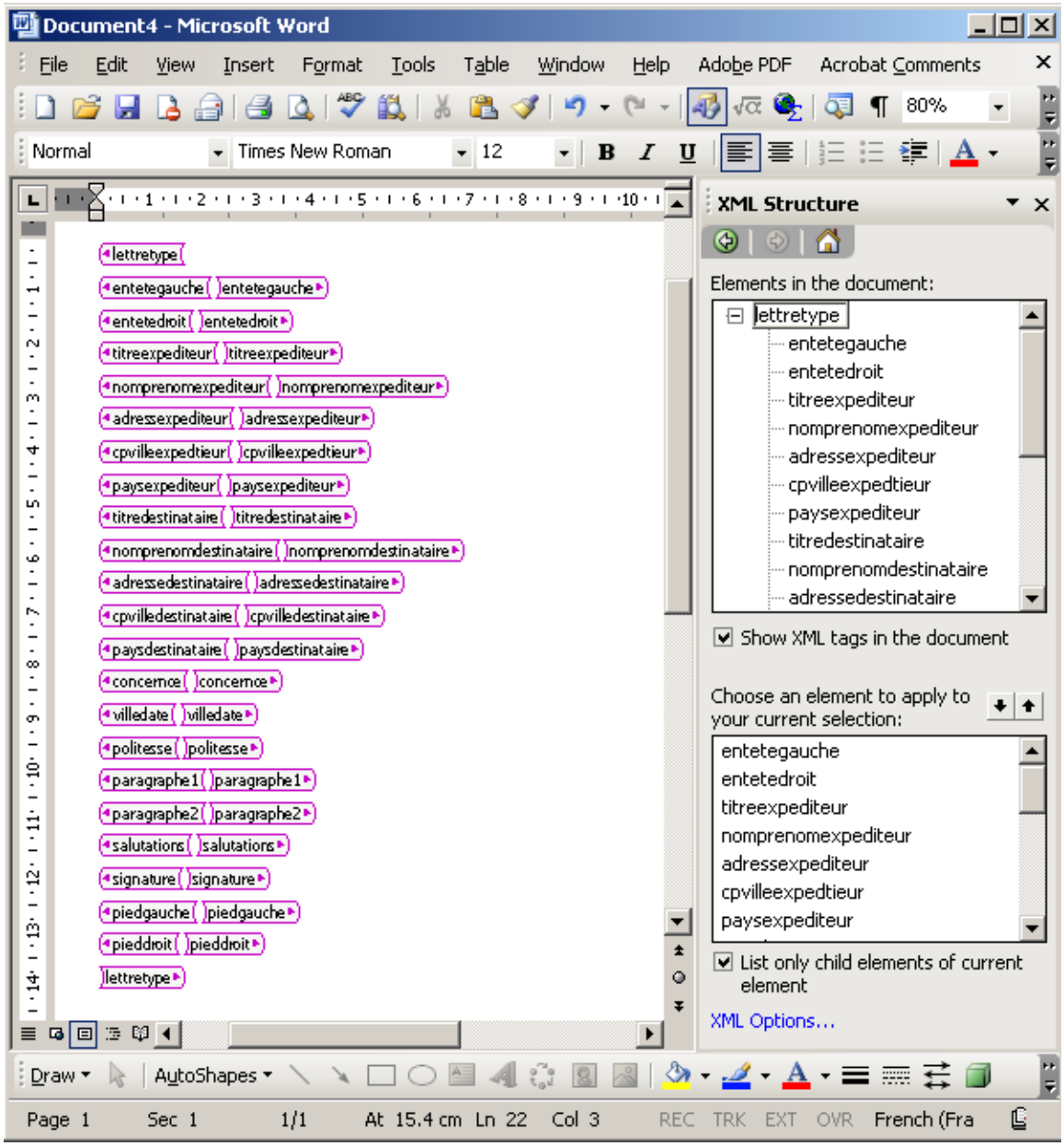


Laissez toujours toutes les cases (*Show XML tags in the document*et *List only Child elements of current element*) cochées (cela ne sert pas vraiment de les décocher).

Maintenant, si le document qui se trouve à gauche du volet Office est vide ou contient quelque chose (peu importe). Cliquez sur *lettretype* après avoir cliqué à l'endroit du document où vous voulez insérer/exporter les futures données du futur fichier XML. Quand vous cliquez sur *lettretype*, une fois sa balise insérée, tous ses neouds enfants doivent apparaître en bas dans le volet Office (voir page suivante):



Maintenant il faut vous positionner dans le document MS Word à gauche de l'écran entre les balises de *lettre type* et cliquer un par un (faites de retour à la ligne au besoin pour que cela soit plus lisible) sur les nœuds que l'on souhaite avoir dans le futur fichier XML. Ainsi (voir page suivante):



Peut-être comprenez-vous déjà l'intérêt de cette méthode ? Si non, voici quelque petites explications:

Rappelez-vous que dans notre premier exemple, notre employé pouvait saisir les données XML et lui appliquer une mise en page prédéfinie. Eh bien ici c'est à peu près pareil à la différence que l'employé aimerait faire un document non standard avec une mise en page personnelle sans avoir à connaître XSL ou CSS.

Comment procède cet employé alors ?:

1. Il renseigne son responsable de la documentation sur le contenu du document

2. Le responsable de la documentation crée un fichier XSD type représentatif de la structure du futur document de son collègue et l'envoie par e-mail (ou le pose sur un disque réseau) à ce dernier.

3. L'employé fait la maquette de son document dans MS Word (mise en page, etc) et place les "balises types" là où il le veut dans son document mais tout en respectant la hiérarchie du XSD.

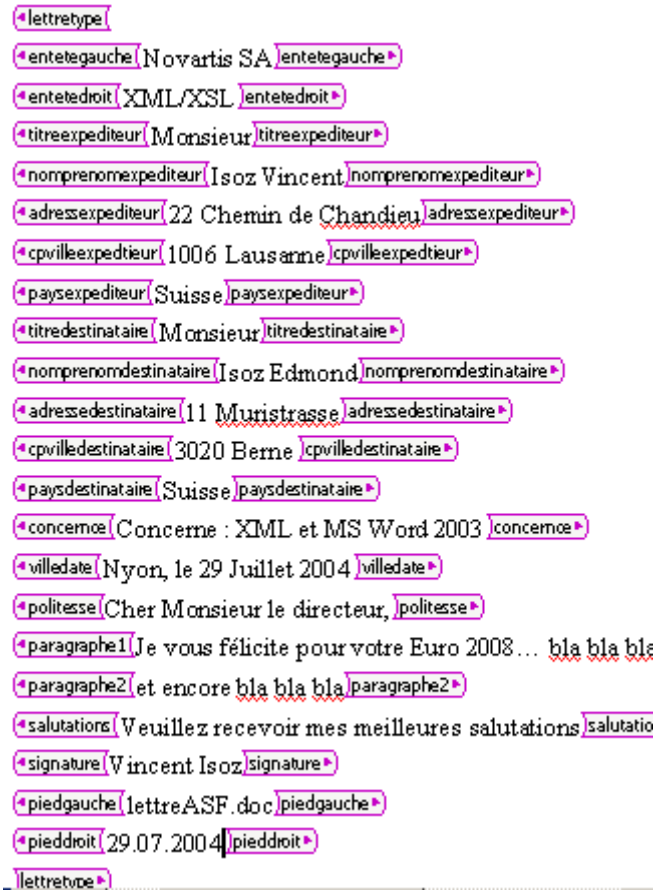
4. Il saisit les données entre les balises. Si vous vous rappelez que dans les XSD nous pouvons définir quelles types de données sont autorisées pour des balises précises, cela nous permet d'avoir un excellent contrôle de ce que fait l'employé.

Remarques:

R1. Dans notre exemple, nous n'avons fait aucune mise en page et nous avons simplement mis le champs les uns à côtés des autres mais faites comme cela vous chante.

R2. Vous aurez des traits violents à l'écran "_____" tant que vous ne respecterez pas la structure du fichier XSD ou le type de données saisies dans les balises. Une fois celle-ci correctement mise en place, tous les traits disparaissent (c'est pas beau ça ? après le correcteur d'orthographe, le correcteur de gramme, voici le correcteur XSD).

Saisissons maintenant des données entre les balises (à nouveau je rappelle que vous pouvez faire une mise en page plus complexe):



L'employé peut ensuite (si… si) enregistrer ce document au format MS Word *.doc (comme un document normal). S'il l'imprime, cela sort comme un document normal (les balises ne sont pas imprimées). Si les balises dérangent l'employé lors du visionnement du document, alors c'est seulement maintenant qu'il est intéressant de cliquer sur la case:

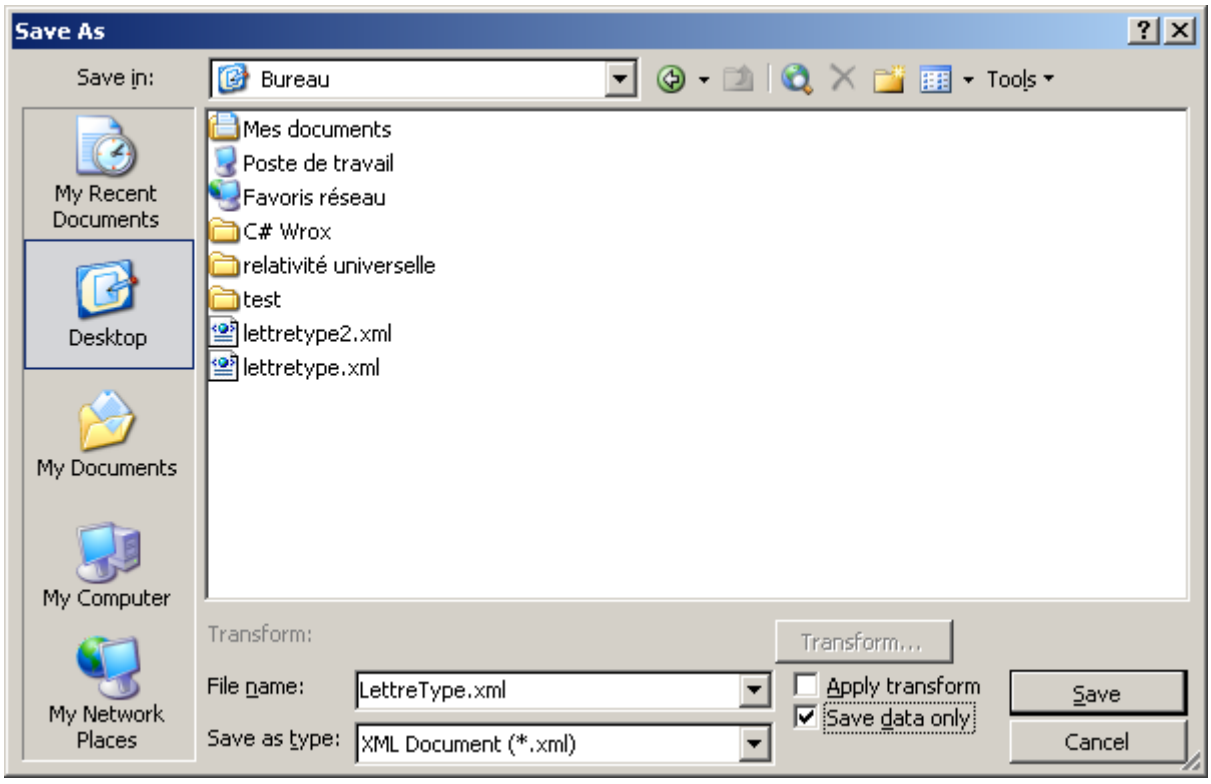
Cela permet d'avoir une vue plus aérée de document.

Et puis maintenant quel est l'intérêt ? Eh bien c'est simple:

1. Déjà l'employé a un document conforme à la charte d'entreprise de par l'usage de l'XSD obtenu par le responsable de la documentation

2. Dans l'objectif d'avoir une GED complète, le responsable de la GED va demander à son collègue de lui envoyer le fichier XML résultant de l'utilisation du fichier XSD et c'est là que c'est nouveau:

Comment l'employé va-t-il envoyé un fichier XML propre (contrairement à Office XP) au responsable de la GED ? Eh bien c'est très simple. Etant donné qu'il a utilisé le XSD il n'a plus qu'à faire les manipulations suivantes: *Enregistrer sous* et:



et choisir un document du type *.xml et de cocher *Save data only* et voici le résultat parfait (à envoyer à la GED):

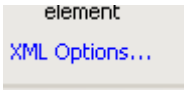
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<lettretype>
    <entetegauche>Novartis SA</entetegauche>
    <entetedroit>XML/XSL</entetedroit>
    <titreexpediteur>Monsieur</titreexpediteur>
    <nomprenomexpediteur>Isoz Vincent</nomprenomexpediteur>
    <adressexpediteur>22 Chemin de Chandieu</adressexpediteur>
    <cpvilleexpedtieur>1006 Lausanne</cpvilleexpedtieur>
    <paysexpediteur>Suisse</paysexpediteur>
    <titredestinataire>Monsieur</titredestinataire>
    <nomprenomdestinataire>Isoz Edmond</nomprenomdestinataire>
    <adressedestinataire>11 Muristrasse</adressedestinataire>
    <cpvilledestinataire>3020 Berne</cpvilledestinataire>
```

```
    <paysdestinataire>Suisse</paysdestinataire>
    <concernce>Concerne: XML et MS Word 2003</concernce>
    <villedate>Nyon, le 29 Juillet 2004</villedate>
    <politesse>Cher Monsieur le directeur,</politesse>
    <paragraphe1>Je vous félicite pour votre Euro 2008… bla bla bla</paragraphe1>
    <paragraphe2>et encore bla bla bla</paragraphe2>
    <salutations>Veuillez recevoir mes meilleures salutations</salutations>
    <signature>Vincent ISOZ</signature>
    <piedgauche>lettreASF.doc</piedgauche>
    <pieddroit>29.07.2004</pieddroit>
</lettretype>
```

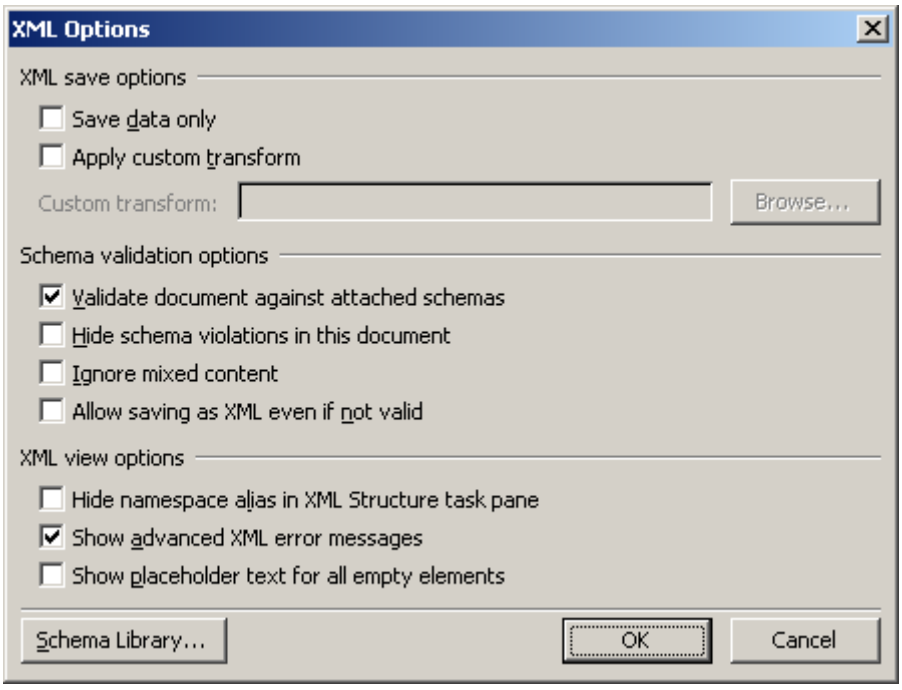
Voilà comment on devrait donc travailler dans toute entreprise digne de ce nom… et également comment devrait travailler tout utilisateur "expert" de MS Word.

Le dernier fichier XML crée peut ensuite être ouvert dans MS Word et nous pouvons à nouveau lui appliquer tout ce que nous avons fait dans l'exemple 1. C'est donc vraiment vraiment très… vous savez quoi…

Voyons quelques détails concernant le XML et le volet Office. Vous vous rappelez de ce petit lien en bas à droite du volet ?:

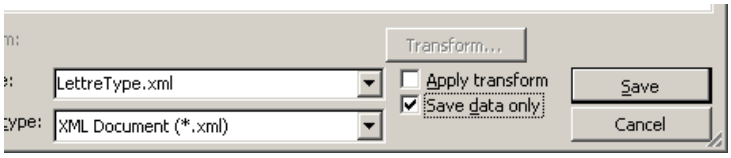


Si vous cliquez dessus, la fenêtre suivante apparaît:



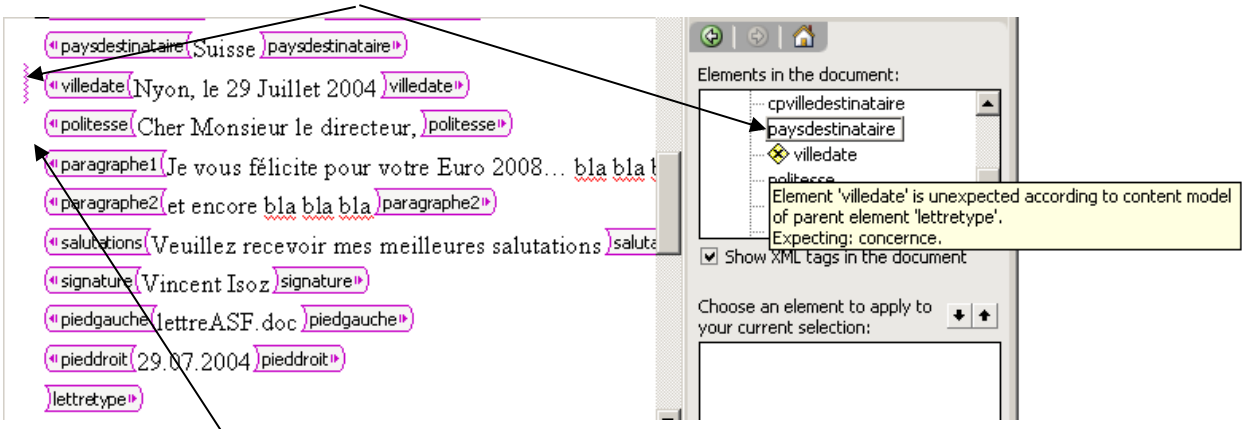
Quelques petites explications:

- *Save Data only* et *Apply custom tranform*. Cocher ces deux cases revient strictement au même à les cocher lorsqu'on sauve (*Enregistrer sous*) notre document au format XML (cette méthode permet simplement de les activer par défaut pour tout futur enregistrement):

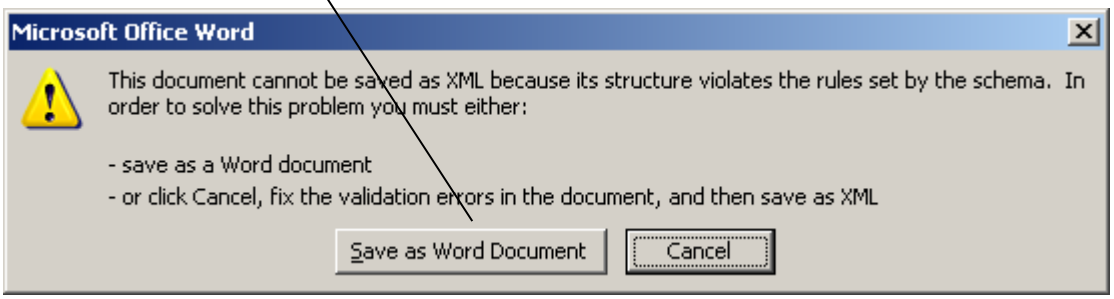


- *Validate document against attached schemas*. Même fonctionnalité que pour MS Excel 2003 lors de l'enregistrement du document en XML: affiche un message d'erreur lors de l'enregistrement en XML si le document MS Word n'est pas conforme au XSD.

  Exemple: nous supprimons dans le document MS Word la balise *Concerne*. Or dans le fichier XSD, nous avons spécifié que cette balise est obligatoire et non facultative donc si nous la supprimons nous avons dans un premier temps deux indications d'erreurs (avant même d'enregistrer en XML):



  et si nous avons activé *Validate document against attached schemas* et que nous essayons d'enregistrer au format XML voici ce qui apparaît à l'écran:

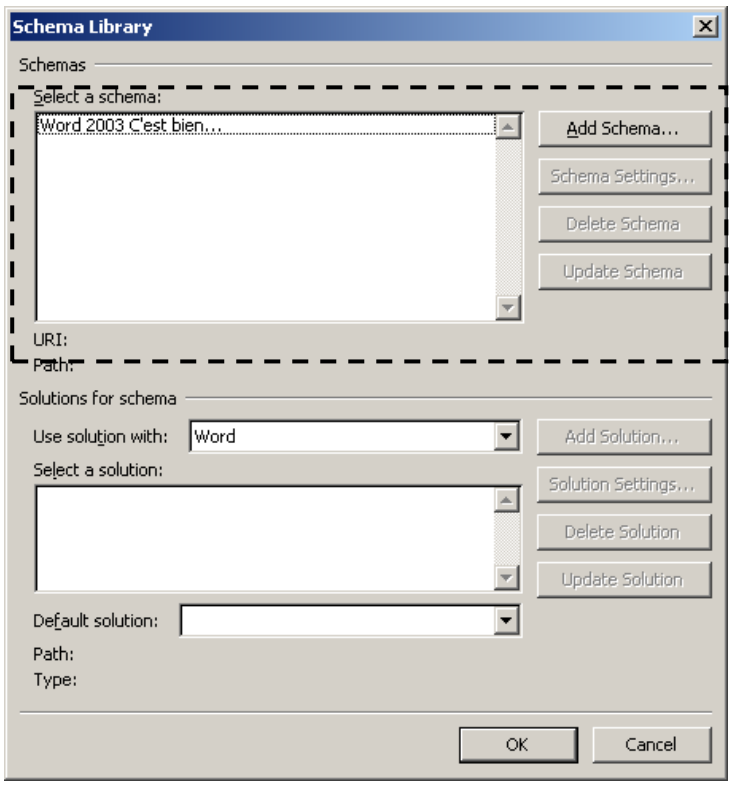


  Donc vous l'aurez compris… vous n'avez pas beaucoup d'alternatives.

- *Hide schema violations in this document*. Masque les lignes violettes de mise en évidence de non conformité avec le XSD. C'est tout…
- *Ignore mixed content.* On en verra l'utilité plus loin dans un exercice.
- *Allow saving as XML file even if not valid.* Ben tout est dit…
- *Hide namespace alias in XML Structure pan.* Ben tout est dit… mais ne sert à rien à ma connaissance car de toute façon ne ne voit pas le namespace…
- *Show advanced error message*. Afficher des messages d'erreurs avancés dans les infobulles:

- *Show placeholder text for all empty elements.* J'ai pas encore trouvé à quoi cela servait… désolé.

Enfin, si vous cliquez sur le bouton *Schema Library* dans le coin inférieur gauche de la fenêtre voici ce qui apparaît:



La partie supérieure de la fenêtre est simple à comprendre. Elle permet simplement d'ajouter/supprimer ou redéfinir la liste des XSD importés (il suffit de cliquer sur les boutons correspondant à droite pour comprendre).

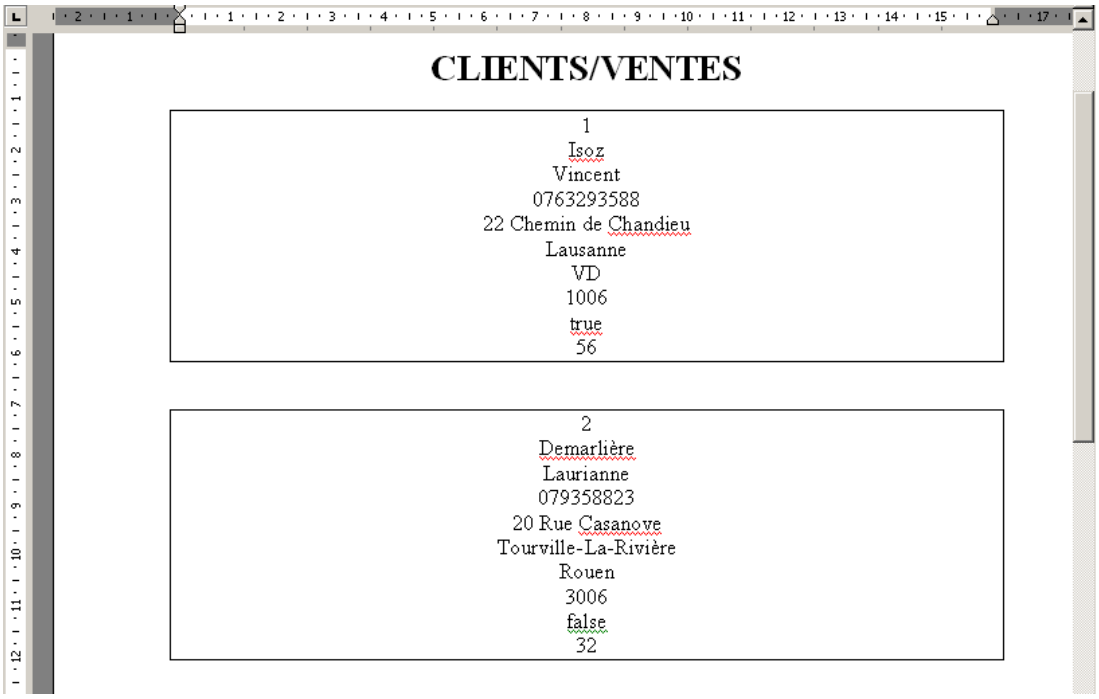
La partie inférieure de la fenêtre (*Solutions for schema*) m'est inconnue. Désolé

On peut également importer ou exporter avec VBA le XML. Les algorithmes sont les mêmes que pour Excel à part qu'on préférera travailler avec des bookmarks plutôt qu'avec des cellules…

10.2.3 Séries de données

MS Word avec le XML peut être aussi utilisé comme outil de saisie de données en masse. Données, qui peuvent ensuite être envoyées dans un tableau comme MS Excel par import XML. Voyons un exemple.

Considérons le document suivant:



CLIENTS/VENTES

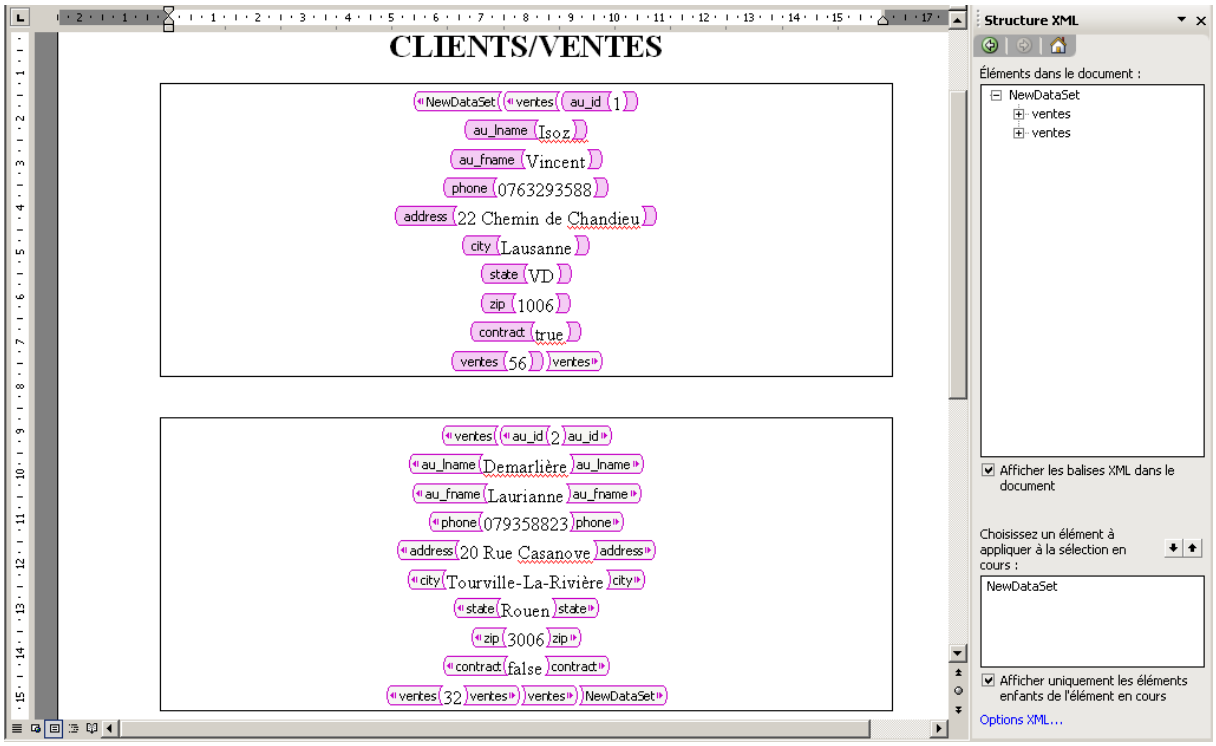
1
Isoz
Vincent
0763293588
22 Chemin de Chandieu
Lausanne
VD
1006
true
56

2
Demarlière
Laurianne
079358823
20 Rue Casanove
Tourville-La-Rivière
Rouen
3006
false
32

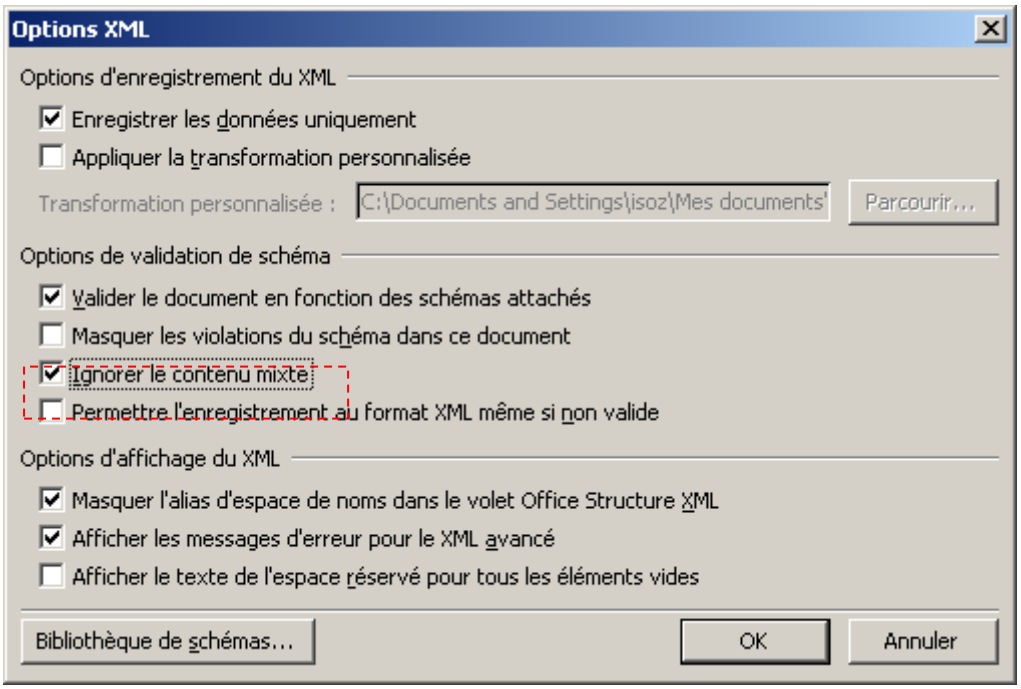
Avec le fichier XSD ci-dessous:

```
<?xml version="1.0" standalone="yes"?>
<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by bob denard (Lost Paradise Inc.) -->
<xsd:schema targetNamespace="ventes-details-schema" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xsd:element name="NewDataSet">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="ventes" minOccurs="0" maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="au_id" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="au_lname" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="au_fname" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="phone" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="address" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="city" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="state" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="zip" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="contract" type="xsd:boolean" minOccurs="0"/>
                            <xsd:element name="ventes" type="xsd:integer"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

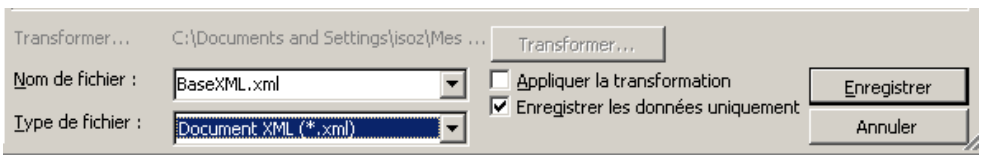
Il est alors possible d'utiliser le schéma pour reconnaître les données comme nous l'avons fait dans les deux exemples précédents:



Attention !!! Si vous souhaitez écrire du texte pour l'esthétique en-dehors et entre les balises xml de namespace il vous faudra alors dans les *Options XML* cocher la case ci-dessous si vous souhaitez pouvoir ensuite enregistrer vos données en XML pur !



Ensuite nous pouvons enregistrer le fichier au format XML (données seulement) sans transformation:



Ce qui donne:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<NewDataSet xmlns="ventes-details-schema">
    <ventes>
        <au_id>1</au_id>
        <au_lname>Isoz</au_lname>
        <au_fname>Vincent</au_fname>
        <phone>0763293588</phone>
        <address>22 Chemin de Chandieu</address>
        <city>Lausanne</city>
        <state>VD</state>
        <zip>1006</zip>
        <contract>true</contract>
        <ventes>56</ventes>
    </ventes>
    <ventes>
        <au_id>2</au_id>
        <au_lname>Demarlière</au_lname>
        <au_fname>Laurianne</au_fname>
        <phone>079358823</phone>
        <address>20 Rue Casanove</address>
        <city>Tourville-La-Rivière</city>
        <state>Rouen</state>
        <zip>3006</zip>
        <contract>false</contract>
        <ventes>32</ventes>
    </ventes>
</NewDataSet>
```

Nous voyons bien ici que le résultat obtenu correspond à ce que nous pouvions attendre de mieux.

10.2.4 WordML

Le lecteur attentif aura cependant remarqué que le XSL standard ne semble cependant pas vraiment adapté à des documents complexes, utilisant des styles et autres concepts de ce genre.

Pour palier à ce problème, Microsoft à mis en place son propre langage de description: le WordML avec ses propres feuilles de styles: le WorsdSL

L'exemple que nous donnons ci-dessous a été pris du site www.labo-dotnet.com (Nicolescu Matthieu). Soit le fichier XML ci-dessous:

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="Transform.xsl"?>
<Students>
    <Student>
        <FirstName>Matthieu</FirstName>
        <LastName>Nicolescu</LastName>
        <Promotion>2004</Promotion>
        <Age>22</Age>
    </Student>
    <Student>
        <FirstName>Eric</FirstName>
        <LastName>Dupont</LastName>
        <Promotion>2007</Promotion>
        <Age>20</Age>
    </Student>

</Students>
```

Lorsque nous l'ouvrons dans MS Word 2003, le résultat nous est connu sans surprises, les balises XML apparaîssent (jusque là donc rien de spécial). Ce qui change c'est le fichier XSL qui maintenant en appliquant le code propriétaire Microsoft WordML (on y voit un namespace!) sera:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <w:wordDocument xmlns:w="http://schemas.microsoft.com/office/word/2003/wordml">
            <w:body>
                <w:p>
                    <xsl:for-each select="Students/Student">
                        <w:r>
                            <w:rPr>
                                <w:b w:val="on" />
                            </w:rPr>
                            <w:t>Nom Complet: </w:t>
                        </w:r>
                        <w:r>
                            <w:t>
                                <xsl:value-of select="FirstName" />
                                <xsl:value-of select="LastName" />
                            </w:t>
                            <w:br />
                        </w:r>
                    </xsl:for-each>
                </w:p>
            </w:body>
        </w:wordDocument>
    </xsl:template>

</xsl:stylesheet>
```

Comme le lecteur peut le voir il y a de nombreuses balises *w* (signifiation *Word* très probablement) qui mettent en forme le texte d'une manière spécifique (on applique cependant le WordSL de la même manière qu'un XSL dans MS Word 2003).

Le langage WordML, donc propre à MS Word 2003 est très conséquent et impossible à traiter dans un ouvrage de moins d'une bonne centaines de pages uniquement consacré à MS Word 2003. Le lecteur qui souhaitera cependant en savoir plus sur les balises MS Word pourra cependant télécharger l'aide complète des schémas XML Word, Excel, Visio, InfoPath 2003 à l'adresse suivante:

http://www.microsoft.com/downloads/details.aspx?FamilyID=fe118952-3547-420a-a412-00a2662442d9&DisplayLang=en

Cependant, on peut très bien se soustraire à cette aide en s'aidant en rédigeant un petit document dans MS Word 2003 (de manière tout à fait standard) et en enregistrant celui-ci au format WordML. Ainsi, on peut apprendre seul.

Quelques petits exemples cependant:

E1. On écrit "Hello World"

```
<?xml version="1.0"?>
<w:wordDocument xmlns:w="http://schemas.microsoft.com/office/word/2003/wordml">
 <w:body>
  <w:p>
   <w:r>
    <w:t>Hello World !</w:t>
  </w:r>
 </w:p>
 </w:body>

</w:wordDocument>
```

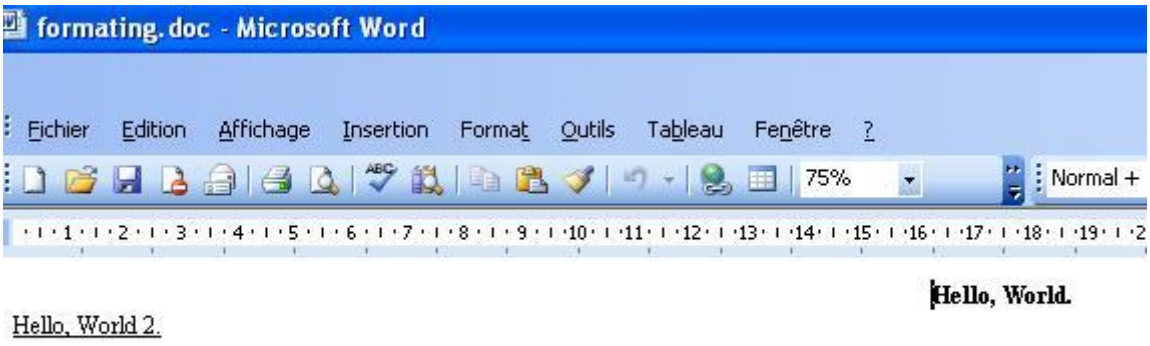
E2. Deux paragraphes:

```
<w:p>
    <w:r>
        <w:t>Hello World 1</w:t>
    </w:r>
</w:p>
<w:p>
    <w:r>
        <w:t>Hello World 2</w:t>
    </w:r>

</w:p>
```

E3. Une compilation:

```
 <w:p>
        <w:pPr>
            <w:jc w:val="center"/>
        </w:pPr>
        <w:r>
            <w:rPr>
                <w:b w:val="on"/>
            </w:rPr>
            <w:t>Hello, World.</w:t>
        </w:r>
   </w:p>
   <w:p>
        <w:r>
            <w:rPr>
                <w:u w:val="single"/>
            </w:rPr>
            <w:t>Hello, World 2.</w:t>
        </w:r>

   </w:p>
```

Qui donne:



Un problème peut se poser à présent: en effet imaginons un document WordML qui contienne un nombre conséquent d'éléments « r ». Il serait fastidieux de recréer à chaque fois les propriétés pour chaque zone de texte sachant qu'on peut considérer que plusieurs zones auront les mêmes propriétés. On va donc utiliser l'élément *styles* qui va nous permettre de définir plusieurs styles et les réutiliser par la suite dans le document. Il faut aussi à faire attention de placer l'élément *styles* avant l'élément *body* dans le document.

E4. Voici un exemple de déclaration de deux styles:

```
<w:styles>
  <w:style w:type="paragraph" w:styleId="MyStyle1" ><w:name w:val="Style1"/>
    <w:pPr>
      <w:jc w:val="center"/>
    </w:pPr>
  </w:style>
  <w:style w:type="character" w:styleId="MyStyle2" ><w:name w:val="Style2"/>
    <w:basedOn  w:val="Italic"/>
    <w:rPr>
      <w:b  w:val="on"/>
    </w:rPr>
  </w:style>

</w:styles>
```

Comme vous pouvez le voir, nous avons deux attributs dans notre élément *style*:

1. L'attribut « type »: Dans notre exemple nous avons mis la valeur à *paragraph* si le style est destiné à un paragraphe (élément *p*) ou à *character* si le style est destiné à une zone de texte (élément *r*).
2. L'attribut « styleID »: qui va nous permettre tout simplement de donner un nom unique à notre style pour pouvoir ensuite l'appeler.

L'appel des différents styles se fait à partir de l'élément « pPr » pour un paragraphe et « rPr » pour une zone de texte. Mais l'élément imbriqué ne sera pas le même: nous allons en effet faire appel à l'élément « pStyle » pour un style destiné à un paragraphe et l'élément « rStyle » pour un style destiné à une zone de texte.

Voici un exemple concret qui se base sur les styles définis ci-dessus:

```
<?xml version="1.0"?>
<w:wordDocument xmlns:w="http://schemas.microsoft.com/office/word/2003/wordml">
    <w:styles>
        <w:style w:type="paragraph" w:styleId="MyStyle1">
            <w:name w:val="Style1"/>
            <w:pPr>
                <w:jc w:val="center"/>
            </w:pPr>
        </w:style>
        <w:style w:type="character" w:styleId="MyStyle2">
            <w:name w:val="Style2"/>
            <w:basedOn w:val="Italic"/>
            <w:rPr>
                <w:b w:val="on"/>
            </w:rPr>
        </w:style>
    </w:styles>
    <w:body>
        <w:p>
            <w:pPr>
                <w:pStyle w:val="MyStyle1"/>
            </w:pPr>
            <w:r>
                <w:t>Hello World !</w:t>
            </w:r>
        </w:p>
        <w:p>
            <w:r>
                <w:rPr>
                    <w:rStyle w:val="MyStyle2"/>
                </w:rPr>
                <w:t>Hello World !</w:t>
            </w:r>
        </w:p>
    </w:body>
</w:wordDocument>
```

Vous pouvez dans votre document WordML spécifier aussi des propriétés tel que le nom de l'auteur, nombre de pages, numéro de version du document. Tout cela se fait à l'aide de l'élément *DocumentProperties*.

Voici un exemple simple attribuant au document le nom de l'auteur, le titre et la version du document:

```
<o:DocumentProperties>
   <o:Title>Exemple Propriete</o:Title>
   <o:Author>Matthieu Nicolescu</o:Author>
   <o:Version>0.1</o:Version>

</o:DocumentProperties>
```

Comme vous pouvez le voir, ici l'élément *DocumentProperties* n'a pas le même préfixe donc n'oubliez pas de le déclarer avec le bon namespace associé dans l'élément root *WordDocument*:

```
<w:wordDocument xmlns:w="http://schemas.microsoft.com/office/word/2003/wordml" xmlns:o="urn:schemas-microsoft-com:office:office">
```

Ce qui donnera le tout mixé ensemble:

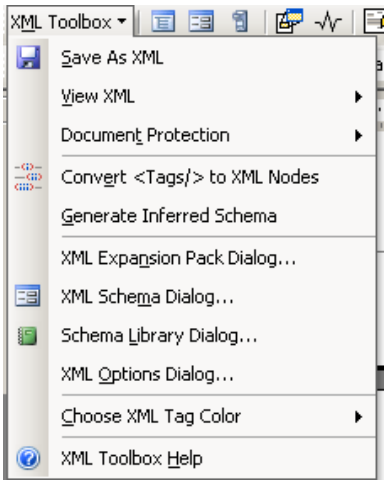
```
<?xml version="1.0"?>
<w:wordDocument xmlns:w="http://schemas.microsoft.com/office/word/2003/wordml" xmlns:o="urn:schemas-
microsoft-com:office:office">
    <o:DocumentProperties>
        <o:Title>Exemple Propriete</o:Title>
        <o:Author>Matthieu Nicolescu</o:Author>
        <o:Version>0.1</o:Version>
    </o:DocumentProperties>
    <w:styles>
        <w:style w:type="paragraph" w:styleId="MyStyle1">
            <w:name w:val="Style1"/>
            <w:pPr>
                <w:jc w:val="center"/>
            </w:pPr>
        </w:style>
        <w:style w:type="character" w:styleId="MyStyle2">
            <w:name w:val="Style2"/>
            <w:basedOn w:val="Italic"/>
            <w:rPr>
                <w:b w:val="on"/>
            </w:rPr>
        </w:style>
    </w:styles>
    <w:body>
        <w:p>
            <w:pPr>
                <w:pStyle w:val="MyStyle1"/>
            </w:pPr>
            <w:r>
                <w:t>Hello World !</w:t>
            </w:r>
        </w:p>
        <w:p>
            <w:r>
                <w:rPr>
                    <w:rStyle w:val="MyStyle2"/>
                </w:rPr>
                <w:t>Hello World !</w:t>
            </w:r>
        </w:p>
    </w:body>
</w:wordDocument>
```

Voilà concernant MS Word 2003 et XML. Nous pouvons faire beaucoup mieux mais un début c'est déjà pas mal.

10.2.4.1 Word XML Toolbox

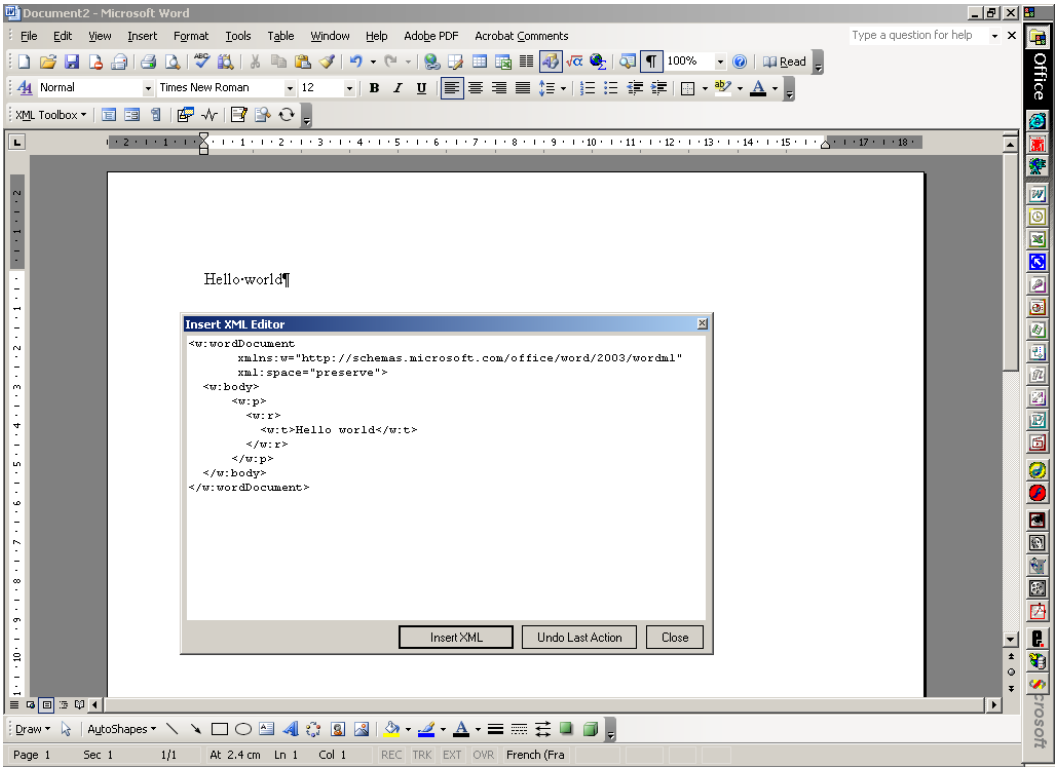
Outre ces outils, Microsoft a développé une petite barre pour MS Word 2003 (nommée "Word XML Toolbox") téléchargable gratuitement sur Internet et qui ressemble à la chose suivante:

Avec dans le menu de la barre d'outils:

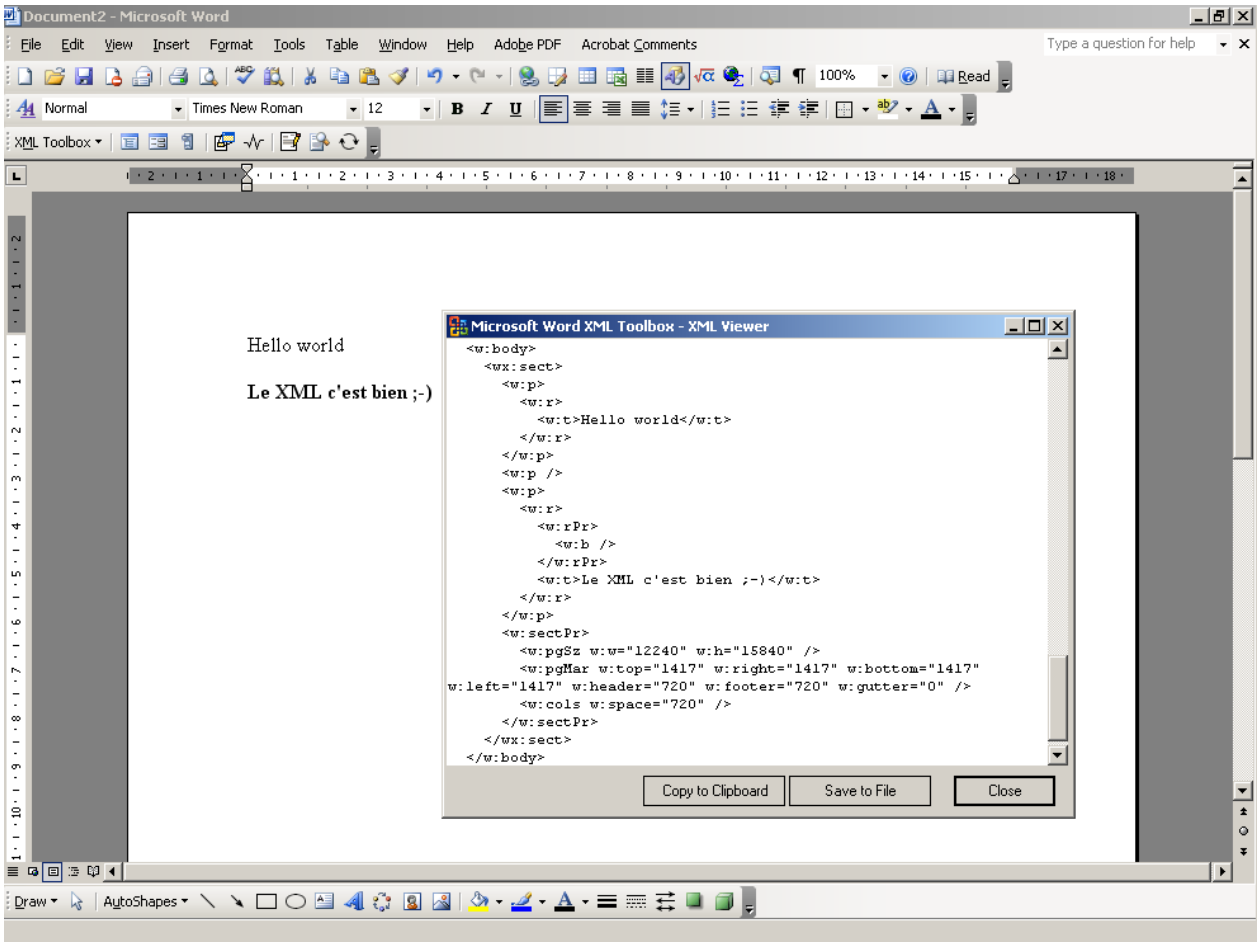


où il n'y a rien vraiment d'extraordinaire.

Ce qui est surtout sympa c'est la boîte XML Dialog qui permet de saisir son texte en WordML tel que le présente la capture d'écran ci-dessous:



ou encore le XML Viewer qui permet de visualiser le WordML de votre texte en cours de rédaction (mais pas en temps réel):



Le reste des options n'a d'intérêt "que" si l'on se trouve dans un document contenant des balises XML.

10.2.5 SmartDocuments

Une grande nouveauté de MS Office 2003 avec XML c'est de pouvoir développer des contenus de documents (MS Excel, MS Word ou encore MS InfoPath) dynamiques à l'aide de la technologie .Net.

Bien que cette technologie s'adresse plutôt à des informaticiens (développeurs) de métier, nous souhaitons quand même montrer un exemple (MS Word 2003) afin que tout utilisateur ait au moins une idée de quoi il s'agit.

Pour cet exemple, nous sommes équipés de Visual Studio.Net 2003, MS Word 2003 (le tout en anglais évidemment) et des exemples fournis par le SDK Microsoft disponibles en téléchargement sur le site Internet de Microsoft à l'adresse suivante:

http://www.microsoft.com/downloads/details.aspx?FamilyID=24a557f7-eb06-4a2c-8f6c-2767b174126f&DisplayLang=en

Pour cet exemple nous disposons des éléments suivants:

1. D'une image nommée: *Alphabet.gif*

2. D'un fichier XML (WordML) nommé *GettysburgAddress.xml*:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?mso-application progid="Word.Document"?>
<w:wordDocument xmlns:w="http://schemas.microsoft.com/office/word/2003/wordml" xmlns:v="urn:schemas-microsoft-com:vml" xmlns:w10="urn:schemas-microsoft-com:office:word"
xmlns:sl="http://schemas.microsoft.com/schemaLibrary/2003/core"
xmlns:aml="http://schemas.microsoft.com/aml/2001/core"
xmlns:wx="http://schemas.microsoft.com/office/word/2003/auxHint" xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:dt="uuid:C2F41010-65B3-11d1-A29F-00AA00C14882" xmlns:st1="urn:schemas-microsoft-com:office:smarttags"
w:macrosPresent="no" w:embeddedObjPresent="no" w:ocxPresent="no" xml:space="preserve">
    <o:SmartTagType o:namespaceuri="urn:schemas-microsoft-com:office:smarttags" o:name="City"/>
    <o:SmartTagType o:namespaceuri="urn:schemas-microsoft-com:office:smarttags" o:name="place"/>
    <o:DocumentProperties>
        <o:Title>Four score and seven years ago our fathers brought forth, upon this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal</o:Title>
        <o:Revision>4</o:Revision>
        <o:TotalTime>1</o:TotalTime>
        <o:Created>2003-08-25T04:40:00Z</o:Created>
        <o:LastSaved>2003-09-12T17:33:00Z</o:LastSaved>
        <o:Pages>1</o:Pages>
        <o:Words>26</o:Words>
        <o:Characters>154</o:Characters>
        <o:Lines>1</o:Lines>
        <o:Paragraphs>1</o:Paragraphs>
        <o:CharactersWithSpaces>179</o:CharactersWithSpaces>
        <o:Version>11.5604</o:Version>
    </o:DocumentProperties>
    <w:fonts>
        <w:defaultFonts w:ascii="Times New Roman" w:fareast="Times New Roman" w:h-ansi="Times New Roman" w:cs="Times New Roman"/>
    </w:fonts>
    <w:styles>
        <w:versionOfBuiltInStylenames w:val="4"/>
        <w:latentStyles w:defLockedState="off" w:latentStyleCount="156"/>
        <w:style w:type="paragraph" w:default="on" w:styleId="Normal">
            <w:name w:val="Normal"/>
            <w:rsid w:val="00BF4B2C"/>
            <w:rPr>
                <wx:font wx:val="Times New Roman"/>
                <w:sz w:val="24"/>
                <w:sz-cs w:val="24"/>
                <w:lang w:val="EN-US" w:fareast="EN-US" w:bidi="AR-SA"/>
            </w:rPr>
        </w:style>
        <w:style w:type="character" w:default="on" w:styleId="DefaultParagraphFont">
            <w:name w:val="Default Paragraph Font"/>
            <w:semiHidden/>
        </w:style>
        <w:style w:type="table" w:default="on" w:styleId="TableNormal">
            <w:name w:val="Normal Table"/>
            <wx:uiName wx:val="Table Normal"/>
            <w:semiHidden/>
            <w:rPr>
                <wx:font wx:val="Times New Roman"/>
            </w:rPr>
            <w:tblPr>
                <w:tblInd w:w="0" w:type="dxa"/>
                <w:tblCellMar>
                    <w:top w:w="0" w:type="dxa"/>
                    <w:left w:w="108" w:type="dxa"/>
                    <w:bottom w:w="0" w:type="dxa"/>
```

```
                    <w:right w:w="108" w:type="dxa"/>
                </w:tblCellMar>
            </w:tblPr>
        </w:style>
        <w:style w:type="list" w:default="on" w:styleId="NoList">
            <w:name w:val="No List"/>
            <w:semiHidden/>
        </w:style>
    </w:styles>
    <w:docPr>
        <w:view w:val="print"/>
        <w:zoom w:percent="100"/>
        <w:removePersonalInformation/>
        <w:doNotEmbedSystemFonts/>
        <w:proofState w:spelling="clean" w:grammar="clean"/>
        <w:attachedTemplate w:val=""/>
        <w:defaultTabStop w:val="720"/>
        <w:punctuationKerning/>
        <w:characterSpacingControl w:val="DontCompress"/>
        <w:optimizeForBrowser/>
        <w:validateAgainstSchema/>
        <w:saveInvalidXML w:val="off"/>
        <w:ignoreMixedContent w:val="off"/>
        <w:alwaysShowPlaceholderText w:val="off"/>
        <w:compat>
            <w:breakWrappedTables/>
            <w:snapToGridInCell/>
            <w:wrapTextWithPunct/>
            <w:useAsianBreakRules/>
            <w:dontGrowAutofit/>
        </w:compat>
    </w:docPr>
    <w:body>
        <wx:sect>
            <w:p>
                <w:r>
                    <w:t>Four score and seven years ago our fathers brought forth, upon this continent, a new nation,
conceived in </w:t>
                </w:r>
                <st1:City w:st="on">
                    <st1:place w:st="on">
                        <w:r>
                            <w:t>Liberty</w:t>
                        </w:r>
                    </st1:place>
                </st1:City>
                <w:r>
                    <w:t>, and dedicated to the proposition that all men are created equal.</w:t>
                </w:r>
            </w:p>
            <w:sectPr>
                <w:pgSz w:w="12240" w:h="15840"/>
                <w:pgMar w:top="1440" w:right="1800" w:bottom="1440" w:left="1800" w:header="720" w:footer="720"
w:gutter="0"/>
                <w:cols w:space="720"/>
                <w:docGrid w:line-pitch="360"/>
            </w:sectPr>
        </wx:sect>
    </w:body>
</w:wordDocument>
```

3. D'un fichier HTML nommé *help.htm*:

```
<html>
    <head></head>
    <body>
        <p>This is help text taken from an <i>external</i> file.</p>
        <ul>
            <li>This is a list from the help file.</li>
            <li>Another list item in the help file.</li>
        </ul>
    </body>
</html>
```

4. D'un fichier XML nommé *managedmanifest.xml:*

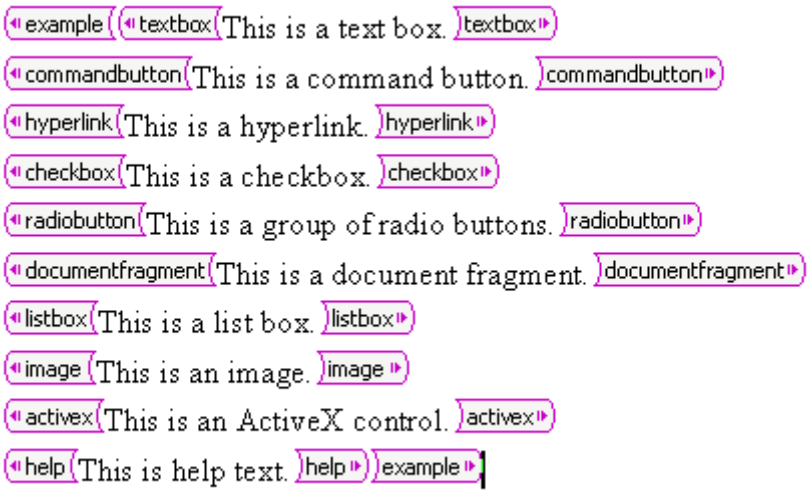
```
<SD:manifest xmlns:SD="http://schemas.microsoft.com/office/xmlexpansionpacks/2003">
    <SD:version>1.1</SD:version>
    <SD:updateFrequency>20160</SD:updateFrequency>
    <SD:uri>SimpleSample</SD:uri>
    <SD:solution>
        <SD:solutionID>SimpleSampleVB7.clsActions</SD:solutionID>
        <SD:type>smartDocument</SD:type>
        <SD:alias lcid="*">Simple Smart Document Sample - VB.NET</SD:alias>
        <SD:file>
            <SD:type>solutionActionHandler</SD:type>
            <SD:version>1.0</SD:version>
            <SD:filePath>bin\SimpleSampleVB7.dll</SD:filePath>
            <SD:CLSNAME>SimpleSampleVB7.clsActions</SD:CLSNAME>
            <SD:managed/>
        </SD:file>
        <SD:file>
            <SD:type>other</SD:type>
            <SD:version>1.0</SD:version>
            <SD:filePath>bin\Interop.MSACAL.dll</SD:filePath>
        </SD:file>
        <SD:file>
            <SD:type>other</SD:type>
            <SD:version>1.0</SD:version>
            <SD:filePath>bin\Interop.ShDocVw.dll</SD:filePath>
        </SD:file>
    </SD:solution>
    <SD:solution>
        <SD:solutionID>schema</SD:solutionID>
        <SD:type>schema</SD:type>
        <SD:alias lcid="*">Simple Smart Document Sample Schema</SD:alias>
        <SD:file>
            <SD:type>schema</SD:type>
            <SD:version>1.0</SD:version>
            <SD:filePath>SimpleSample.xsd</SD:filePath>
        </SD:file>
    </SD:solution>
</SD:manifest>
```

5. D'une deuxième image nommé *simplesample.bmp*

6. D'un schéma XSD nommé *SimpleSample.xsd*:

```
<xsd:schema targetNamespace="SimpleSample" xmlns="SimpleSample"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xsd:complexType name="exampleType">
        <xsd:all>
            <xsd:element name="textbox" type="xsd:string"/>
            <xsd:element name="commandbutton" type="xsd:string"/>
            <xsd:element name="help" type="xsd:string"/>
            <xsd:element name="radiobutton" type="xsd:string"/>
            <xsd:element name="checkbox" type="xsd:string"/>
            <xsd:element name="listbox" type="xsd:string"/>
            <xsd:element name="image" type="xsd:string"/>
            <xsd:element name="documentfragment" type="xsd:string"/>
            <xsd:element name="activex" type="xsd:string"/>
            <xsd:element name="hyperlink" type="xsd:string"/>
        </xsd:all>
    </xsd:complexType>
    <xsd:element name="example" type="exampleType"/>
</xsd:schema>
```

7. D'un document MS Word 2003 nommé *SimpleSample.doc* dans lequel on a préparé la structure suivante s'inspirant bien évidemment du XSD (voir page suivante):

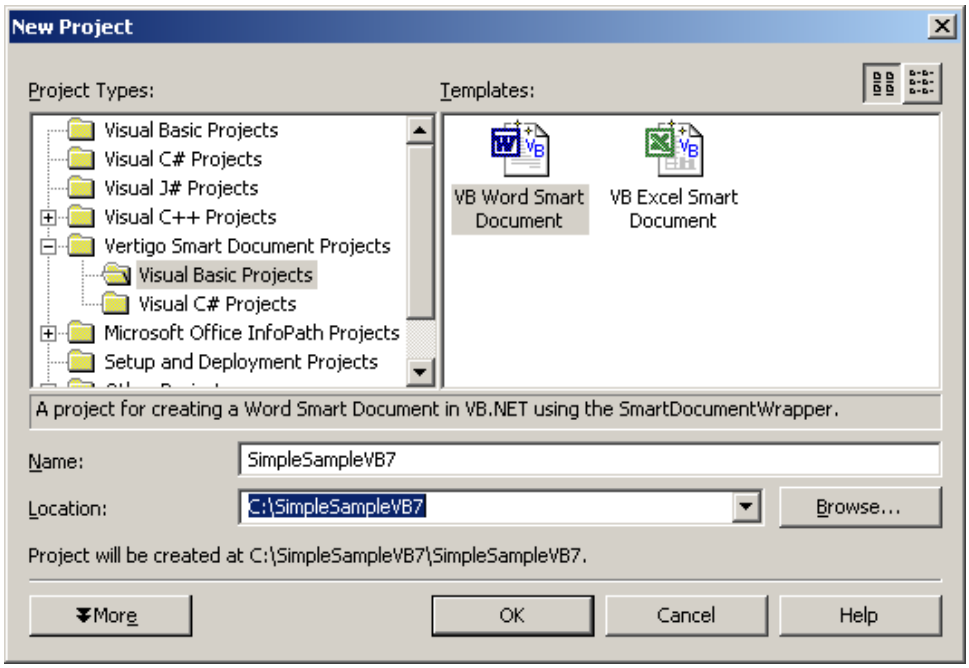


L'objectif: Nous allons créer un projet VB.Net (nommé *SimpleSampleVB7.sln*) dans Visual Studio.Net pour que l'utilisateur de ce document MS Word 2003 puisse saisir dynamiquement le contenu des balises XML à l'aide du Volet de MS Office 2003.

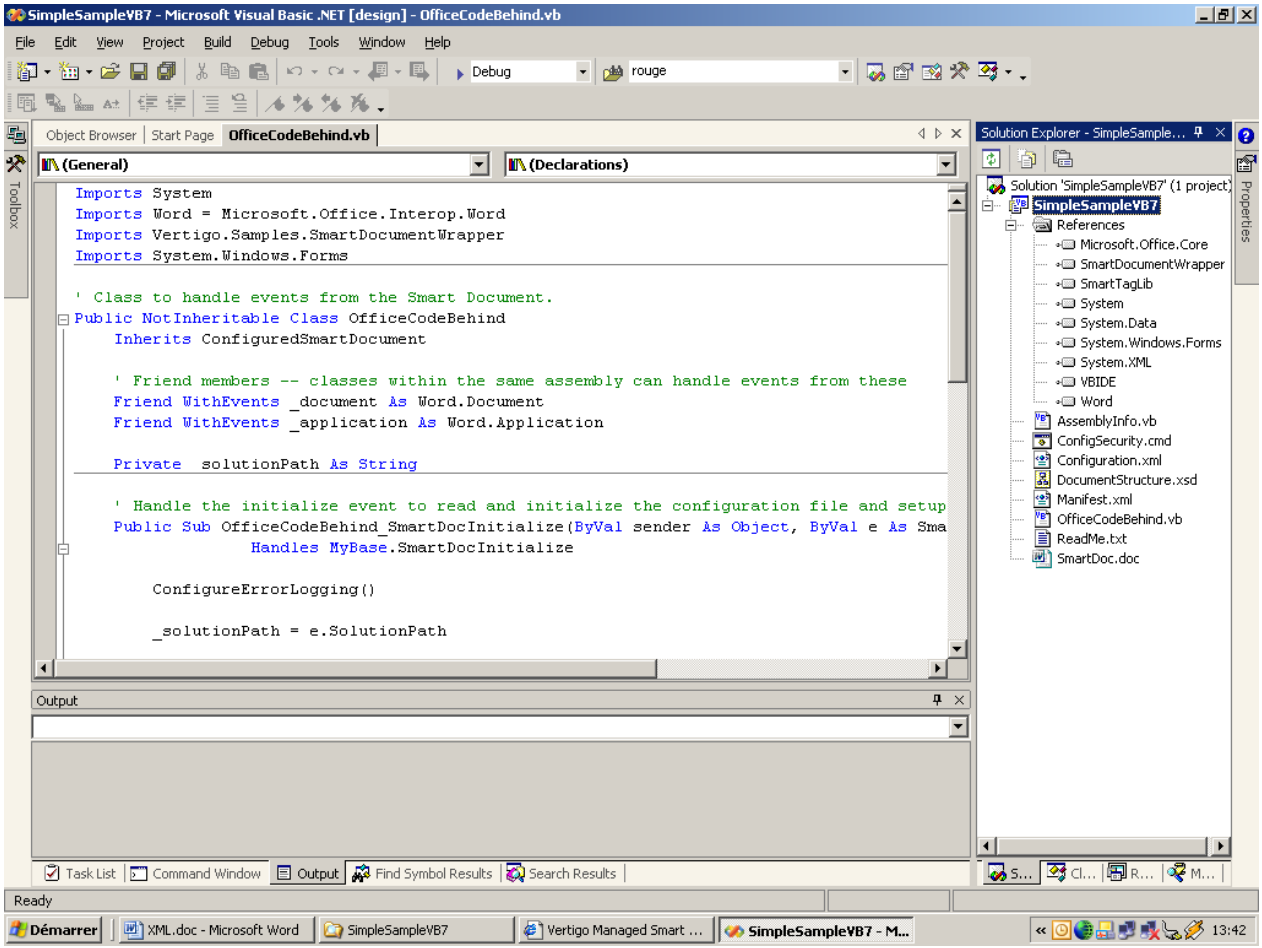
Pour créer ce projet nous avons au préalable télécharge le module gratuit de développement de la société Vertigo disponible à l'adresser suivante:

http://www.vertigosoftware.com/wrapper.htm

Une fois ce module téléchargé et installé, quand nous ouvrons MS Visual Studio.Net 2003 nous avons la possibilité de créer un projet un projet MS Word 2003 VB.Net ou C# selon loisir:



Une fois ce projet crée, nous avons tout ce qu'il faut (les références, le XSD, le XML) qui est généré (voir le *Solution Explorer* à droite de la capture d'écran) pour pouvoir comment à travailler:



10.2.6 SmartTags

Une autre grande nouveauté de MS Office 2003 avec XML (et ne parlons pas de la version 2007…) c'est aussi de pouvoir utiliser et développoer les "SmartTags".

Bien que cette technologie s'adresse plutôt à des informaticiens (développeurs) de métier, nous souhaitons quand même montrer un exemple (MS Word) afin que tout utilisateur ait au moins une idée de quoi il s'agit.

Nous allons ici utilser un exemple donné par un internaute. Il propose de télécharger le fichier *LOGFILE.XML* suivant (attention à le télécharger en UniCode!) et de l'installer dans MS Word:

```
<FL:smarttaglist xmlns:FL="urn:schemas-microsoft-com:smarttags:list">
    <FL:name>Whois IP Lookup</FL:name>
    <FL:lcid>1033,0</FL:lcid>
    <FL:description>Lookup a domain name.</FL:description>
    <FL:smarttag type="urn:schemas-microsoft-com:office:smarttags#whois">
        <FL:caption>Whois IP Lookup</FL:caption>
        <FL:re>
            <FL:exp>\b(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\b
        </FL:exp>
        </FL:re>
        <FL:actions>
            <FL:action id="WhoisNA">
                <FL:caption>Find Whois for IP North America</FL:caption>
                <FL:url>http://ws.arin.net/cgi-bin/whois.pl?queryinput={TEXT}</FL:url>
            </FL:action>
            <FL:action id="WhoisEurope">
                <FL:caption>Find Whois for IP Europe</FL:caption>
                <FL:url>http://www.ripe.net/perl/whois?searchtext={TEXT}</FL:url>
            </FL:action>
            <FL:action id="WhoisAsia">
                <FL:caption>Find Whois for IP Asia</FL:caption>
                <FL:url>http://www.apnic.net/apnic-bin/whois.pl?searchtext={TEXT}</FL:url>
            </FL:action>
            <FL:action id="WhoisLatinAmerica">
                <FL:caption>Find Whois for IP Latin America</FL:caption>
                <FL:url>http://lacnic.net/cgi-bin/lacnic/whois?query={TEXT}</FL:url>
            </FL:action>
            <FL:action id="Whatcountry">
                <FL:caption>Find the country</FL:caption>
                <FL:url>http://software77.net/cgi-bin/ip-country/geo-ip.pl?wwwip={TEXT}</FL:url>
            </FL:action>
        </FL:actions>
    </FL:smarttag>
</FL:smarttaglist>
'
```

Nous y voyons une regular expression pour l'adresse IP. Elles ne sont pas simples à déterminer mais utiles dans bon nombre de cas donc InfoPath par exemple. De la documentation sur le sujet est disponible ici:

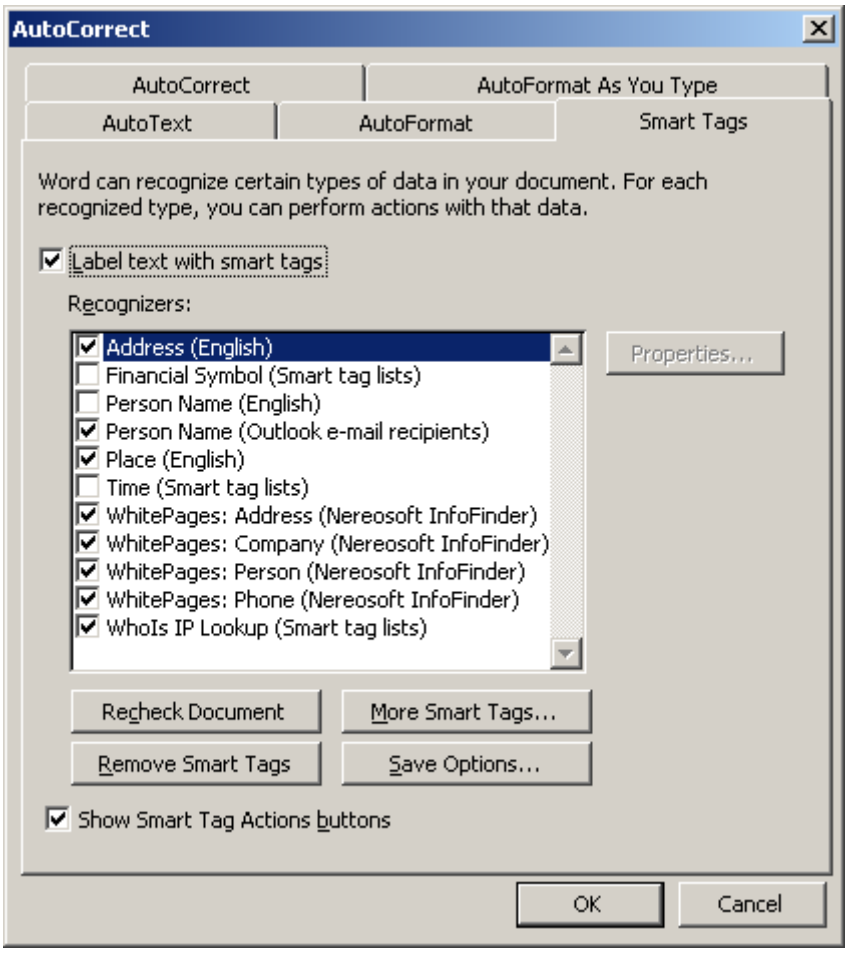
http://www.regular-expressions.info

Les personnes connaissant un peu la programmation reconnaitront que l'exemple fait appel à une requête *Perl* sur une page web sur un serveur externe.

Pour que ce SmartTag fonctionne il faut d'abord fermer MS Word, ensuite copier le fichier XML dans le répertoire suivant:

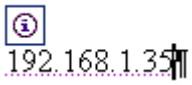
C:\Program Files\Common files\Microsoft Shared\Smart Tag\Lists

Ensuite, il faut dans MS Office prendre garde à avoir les SmartTags activées en allant dans le menu *Tools/AutoCorrect Options…*tel que :



Vous activez ici *WhoIs IP Lookup* et ensuite vous fermez et rouvrez MS Word.

Ainsi, si nous saisissons une adresse IP dans un document MS Word nous avons la chose suivante qui apparaît:



On peut très bien changer les URLs si on le souhaite de même que les captions.

On peut changer également l'expression de reconnaissance par:

```
<FL:exp>[B|b]ug</FL:exp>
```

Ce qui produit la même action si l'utilisateur tape *Bug* ou *bug* ….

Ou si on veut toujours les mêmes actions pour une saisie du genre *Bug* #123456 ou *bug* #123456 on choisir pour expression:

```
<FL:exp>[B|b]ug\s(#)\d{6}</FL:exp>
```

ou \s représente un espace vide.

Ou encore pour les numéros de téléphone américains:

```
<FL:exp>((\(\d{3}\))|(\d{3}-))?\ d{3}-\d{4}</FL:exp>
```

Décortiquons un peu cela… \d{3}-\d{4} signifie que l'expresion doit cotenir exactement trois digits suivi d'un tiret et ensuite exactement de 4 digits

Ensuite pour ((\(\d{3}\))|(\d{3}-))? où le | indique l'opérateur "OR" (comme en LotusScript) signifie que que l'expression \d{3}-\d{4} peut être préfixée par 3 digits supplémentaires avec un tiret (\d{3}-) ou par 3 digits supplémentaires entre paranthèses ((\(\d{3}\)).

Ainsi, notre SmartTag s'executera quand l'utilisateur saisira (281)866-7444 ou 281-866-7444.

Le fait que l'ensemble ((\(\d{3}\))|(\d{3}-)) soit entre paranthèses et suivi d'un point d'interrogation signifie que l'écriture de 866-7444 suffira aussi pour que le SmartTag fonctionne!

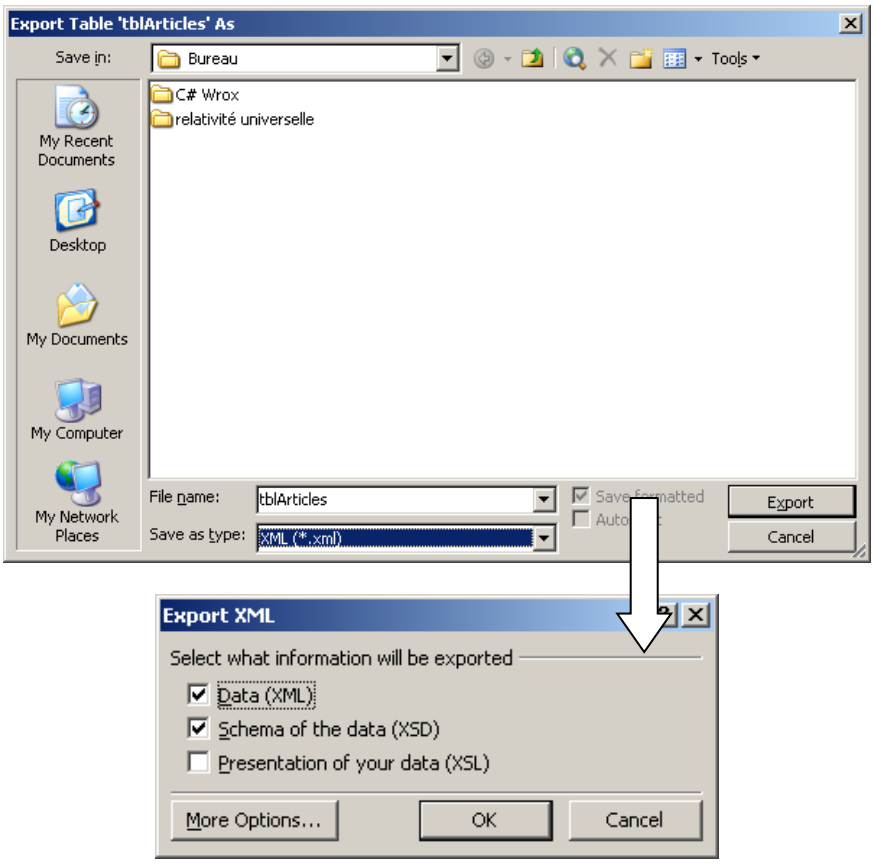
10.3 MS Access 2003

Le méta-langage XML est nous le voyons absolument incontournable aujourd'hui dans les nouvelles technologies et s'impose comme standard international dans l'échange des données. Ainsi, MS Access est-il directement concerné. Cette voie vers le XML que prennent tous les logiciels dans le domaine de l'informatique se fait particulièrement ressentir pour les utilisateurs bureautiques avec la version MS Office 2003 où il est omniprésent même dans MS Word et MS Excel. Il convient dès lors d'acquérir des nouvelles méthodes de travail. Malheureuses, l'inertie d'adaptation des utilisateurs moyen fait que probablement cette technologie sera omniprésente chez le grand publique vers 2010…. voir 2015 (la tendance se confirme au fur et à mesure que les années passent…).

Étant donné qu'un des objectifs principaux de l'XML mis à part tout ce que nous avons dit est la création d'une GED, il nous faut donc savoir importer et exporter des données en XML de ou vers une base de données. Regardons cela avec MS Access 2003.

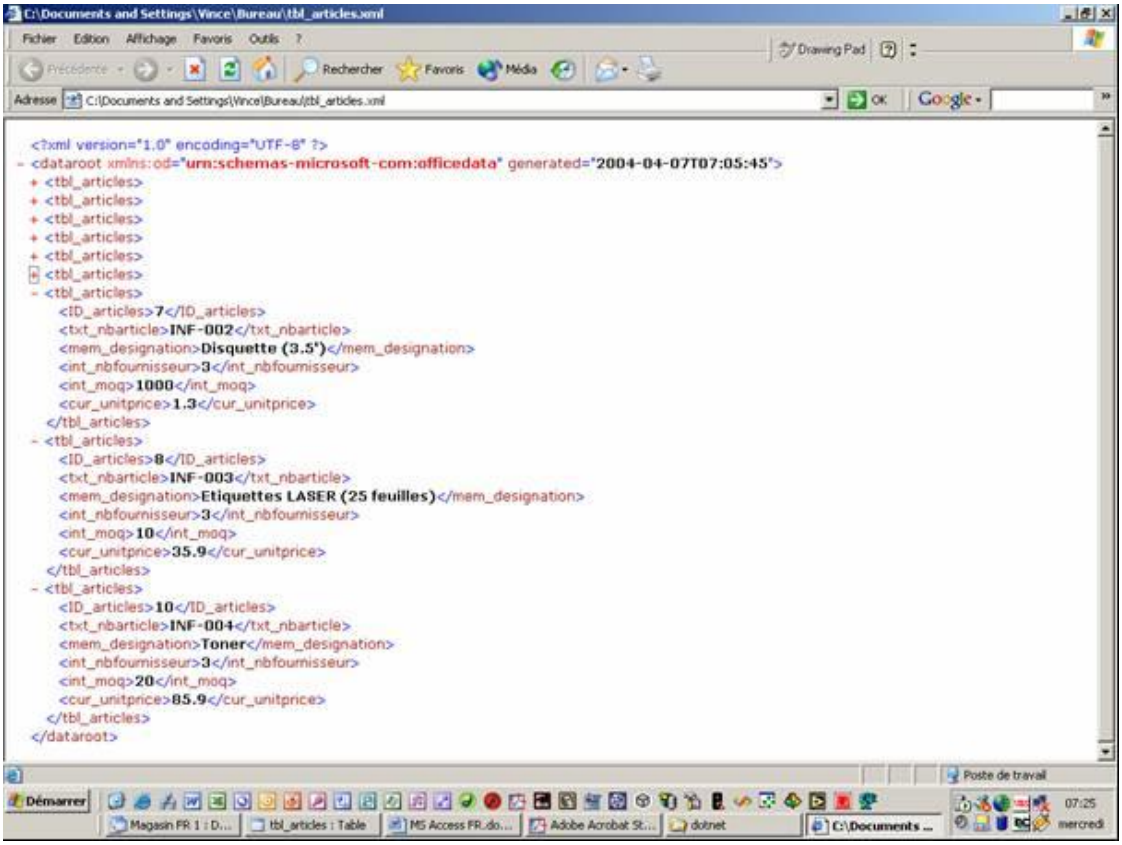
1. Ouvrez une table quelconque dans un base MS Access *.mdb (pour l'exemple nous nous sommes servis d'une table nommée *Articles* dans la pièce jointe téléchargeable)

2. Allez dans le menu *Fichier/Exporter* et choisissez l'option *XML* (si MS Access vous demande de générer un fichier XSL et XSD… laissez tomber !!! Le code y est catastrophique… et c'est inutilisable):



3. Ouvrez le fichier XML résultant dans Internet Explorer

Voici à quoi ressemblera le résultat:



Nous voyons que le code est beaucoup plus propre que dans MS Office 2002 (ce qui est logique par ailleurs).

Une autre manière de générer un fichier XML de façon très modulable à partir de VBA est la suivante (exemple générique pris sur Internet et facilement généralisable). L'exemple ci-dessous créer une table Access et y met des champs et enregistrements et en exporte le contenu dans un fichier XML selon la même technique que pour Excel:

```
Option Compare Database
Option Explicit

'Ne pas oublier d'installer la référence à ActiveX Data Objects 2.0
Sub ExportXML()

    Dim tblRecordset As New ADODB.Recordset

    tblRecordset.Open "tblArticles", CurrentProject.Connection, adOpenKeyset,
    adLockOptimistic

   'on crée le fichier XML
    Open "C:\OutPut.xml" For Output As #1
        Print #1, "<?xml version=""1.0"" encoding=""iso-8859-1""?>"
        Print #1, "<table>"
        Do While Not tblRecordset.EOF
            Print #1, " <record>"
            Print #1, " <code>" & tblRecordset.Fields("strNbArticle") & "</code>"
            Print #1, " <moq>" & tblrecordset.Fields("intMoq") & "</moq>"
```

```
                Print #1, " <fourn>" & tblRecordset.Fields("tblFournisseurNom") &
                "</fourn>"
                Print #1, " </record>"
                tblRecordset.MoveNext
            Loop
            Print #1, "</table>"
        Close #1
        tblRecordset.Close
        Set tblRecordset = Nothing

End Sub
```

Inversement, il est possible d'importer une ficher XML par le code suivant (le nom de la table sera celui du nœud racine du fichier XML):

```
Sub ImportXML()
   Application.ImportXML "c:\tblArticles.xml"
End Sub
```

C'est beau non….

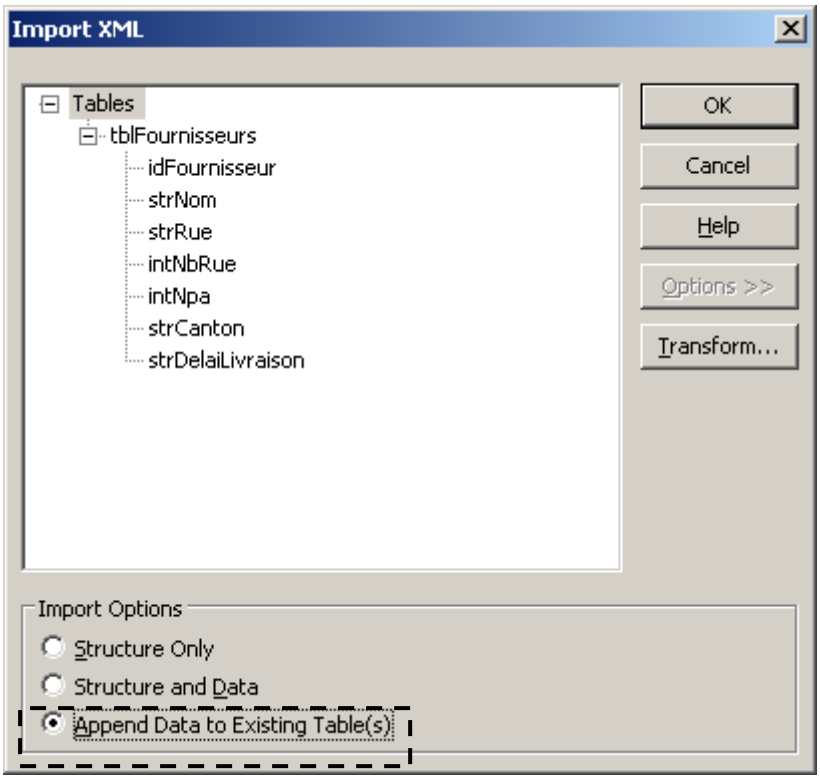
Autre chose pour appliquer un XSL à un document XML exporté et enregistrer le tout dans un nouveau fichier XML (ce code peut être utilisé dans n'importe que logiciel de la suite Office…):

```
Sub XformXML()
    Application.TransformXML _
    DataSource:="C:\Simple.xml", _
    TransformSource:=" C:\XLSStyleSheet.xsl", _
    OutputTarget:="C:\XformedSimple.htm", _
    WellFormedXMLOutput:=False
End Sub
```

Considérons maintenant le fichier *tblFournisseursImport.xml* donné par:

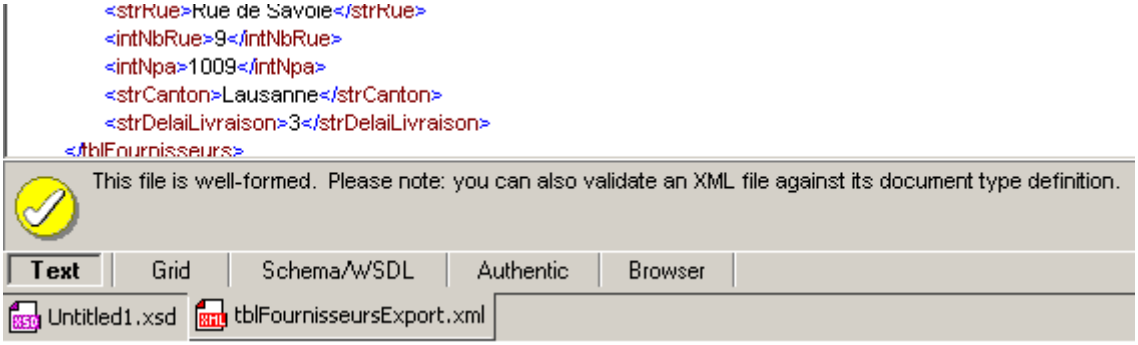
```
<?xml version="1.0" encoding="UTF-8"?>
<dataroot>
    <tblFournisseurs>
        <idFournisseur>4</idFournisseur>
        <strNom>Excelsia SA</strNom>
        <strRue>Rue de Savoie</strRue>
        <intNbRue>9</intNbRue>
        <intNpa>1009</intNpa>
        <strCanton>Lausanne</strCanton>
        <strDelaiLivraison>3</strDelaiLivraison>
    </tblFournisseurs>
    <tblFournisseurs>
        <idFournisseur>5</idFournisseur>
        <strNom>Herdt AG</strNom>
        <strRue>Limatt Strasse</strRue>
        <intNbRue>116</intNbRue>
        <intNpa>3234</intNpa>
        <strCanton>Zürich</strCanton>
        <strDelaiLivraison>8</strDelaiLivraison>
    </tblFournisseurs>
</dataroot>
```

Importons-en les données dans la table *tblFournisseurs* existante (dans le même base *.mdb qu'avant) en utilisant l'assistant manuel *Fichier/Données externes/Importer*:



Exportez maintenant toute la table *tblFournisseur* au format XML sous le nom *tblFournisseurExport*.

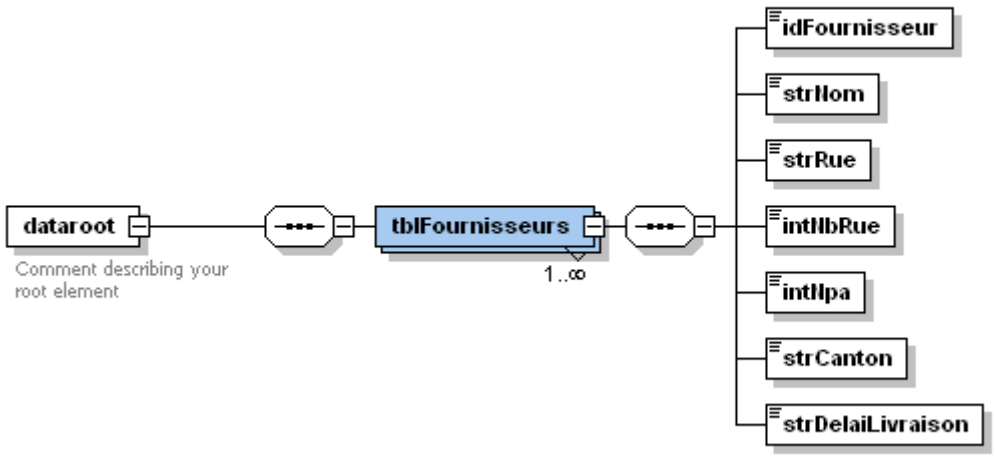
Si nous ouvrons ce fichier XML sortant dans un logiciel spécialisé comme XMLSpy, nous voyons bien que le fichier est conforme (voilà qui est rassurant):



Voici le fichier validateur XSD correspondant (obligatoire pour travailler avec proprement dans MS Excel et MS Word 2003):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="dataroot">
        <xs:annotation>
            <xs:documentation>Comment describing your root element</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="tblFournisseurs" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="idFournisseur" type="xs:integer"/>
                            <xs:element name="strNom" type="xs:string"/>
                            <xs:element name="strRue" type="xs:string"/>
                            <xs:element name="intNbRue" type="xs:integer"/>
                            <xs:element name="intNpa" type="xs:integer"/>
                            <xs:element name="strCanton" type="xs:string"/>
                            <xs:element name="strDelaiLivraison" type="xs:integer"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

et son schéma XMLSpy correspondant:



Il existe maintenant une infinité de possibilité pour mettre en forme notre fichier XML dans MS MS Word 2003 ou Internet Explorer à l'aide d'une feuille de mise en page XSL tel que:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<html>
<body>
    <table border="1" cellspacing="0" cellpadding="3">
        <tr bgcolor="#FFFF00">
            <td>Nom Fournisseur</td>
            <td>Adresse </td>
        </tr>
        <xsl:for-each select="dataroot/tblFournisseurs">
        <xsl:choose>
        <xsl:when test=".[strNom='Infolearn SA']">
        <tr bgcolor="#00FF00">
            <td><xsl:value-of select="strNom"/></td>
            <td><xsl:value-of select="strRue"/>nbsp<xsl:value-of select="intNbRue"/>nbsp<xsl:value-of select="intNpa"/>nbsp
<xsl:value-of select="strCanton"/></td>
        </tr>
        </xsl:when>
        <xsl:otherwise>
        <tr>
            <td><xsl:value-of select="strNom"/></td>
            <td><xsl:value-of select="strRue"/><xsl:value-of select="intNbRue"/><xsl:value-of select="intNpa"/><xsl:value-of
select="strCanton"/></td>
        </tr>
        </xsl:otherwise>
        </xsl:choose>
        </xsl:for-each>
    </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Appliqué au fichier XML, cela devient dans Internet Explorer:

Nom Fournisseur	Adresse
Duplibureau	Ruelle du Soleil 0 2003Neuchâtel
La maison du papier	Rue des Anges 17 1010Lausanne
Tartanpion	Grand-Rue 115 1205Genève
Infolearn SA	Rue de Savoie 9 1009 Lausanne
Herdt AG	Limatt Strasse 116 3234Zürich

Voilà ce sera tout pour notre découverte du monde merveilleux de l'XML avec MS Access 2003.

Bien sûr, XML+Access+Word+Excel+InfoPath 2003 forment à eux 5 dans les outils classiques de la suite professionnelle MS Office un outil d'une puissance, et d'une "compliance" sans égal !

10.4 MS InfoPath 2003

Infopath est un des 6 nouveaux éléments de la suite MS Office System 2003. Il met trivialement en évidence la stratégie de Microsoft de proposer des outils bureautiques performants de gestion et traitement de données au format XML.

MS Infopath est un logiciel de création de formulaires dont les possibilités sont infiniment plus grandes que MS Word et même Adobe Acrobat 6.0.

Nous allons ainsi faire deux exemples. Le premier sera quasi-statique simple ("plat") et utilisera des connaissances accessibles presque à n'importe qui. Le second exemple, complètement dynamique, demandera des connaissances de MS Access.

Remarque: l'utilisation ayant été grandement simplifiée depuis la sortie de son SP1 il est important de savoir que les exemples donnés ci-dessous ont été tirés des captures d'écran de la version MS InfoPath 2003 SP1.

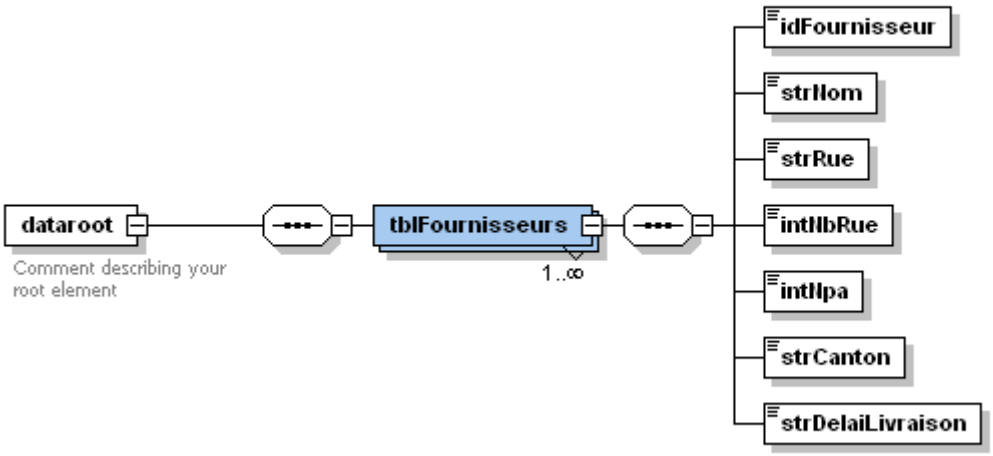
Donc il faut avoir installé le SP1 d'InfoPath

10.4.1 Saisie de données (exemple 1)

Commençons par l'exemple simple. Nous allons créer un formulaire InfoPath qui permet d'ajouter à notre base Access manuellement de nouveaux fournisseurs (nous faisons donc référence aux exemples donnés concernant MS Access 2003 précédemment).

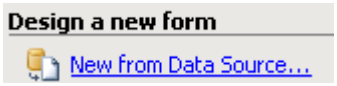
Le but de ce document n'état pas d'apprendre MS InfoPath mais de voir le lien qu'il y a entre lui et le XML, je ne m'étendrai pas sur la manière de créer le formulaire en soi (c'est accessible même a quelqu'un qui débute en informatique).

Mais avant de créer le formulaire, il nous faut le validateur XSD de la table *tblFournisseurs* de notre base *.mdb. Rappelons qu'il est le suivante:

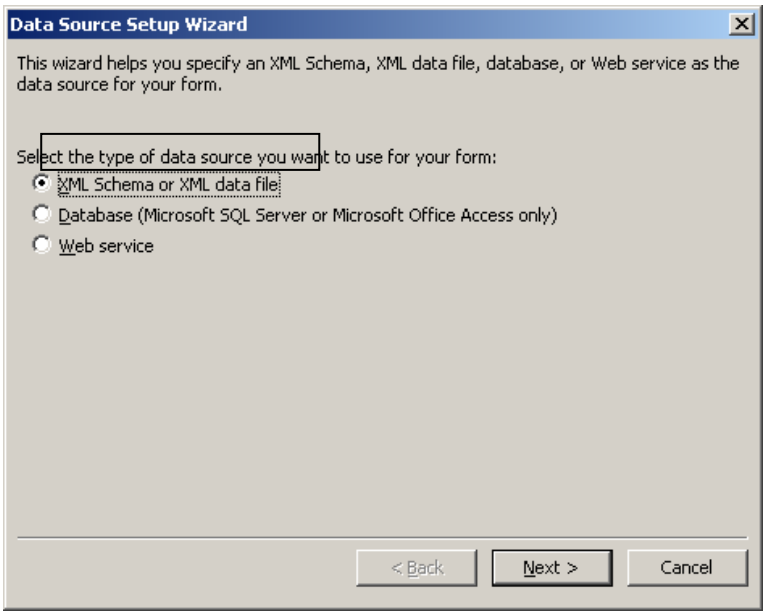


Maintenant, quand vous ouvrez MS InfoPath, dans le volet Office choisissez :

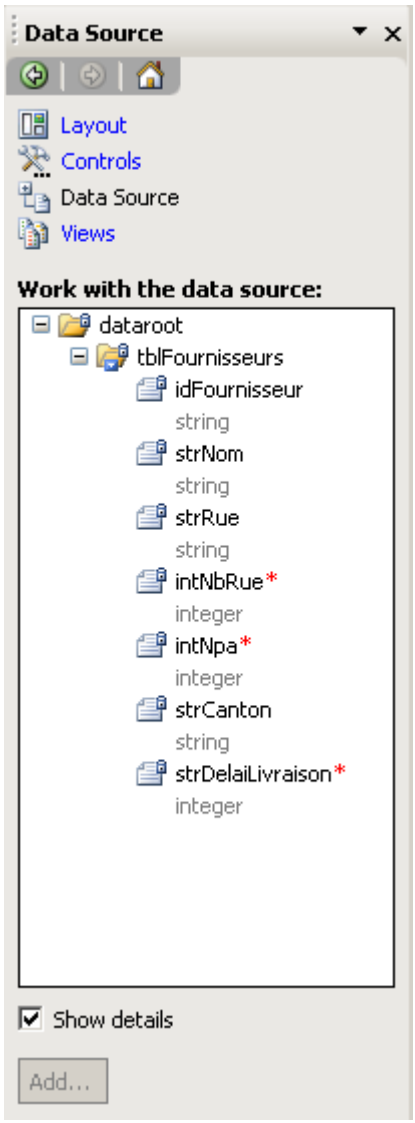
et ensuite toujours dans le volet Office:



Dans la fenêtre qui apparaît sélectionnez:

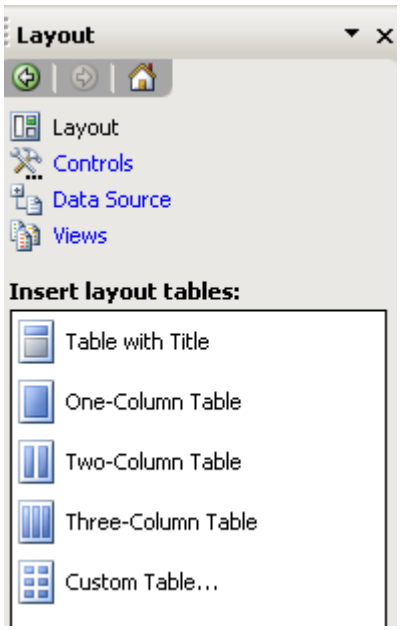


et allez chercher le fichier XSD. Une fois ceci fait vous devriez avoir dans le volet Office:

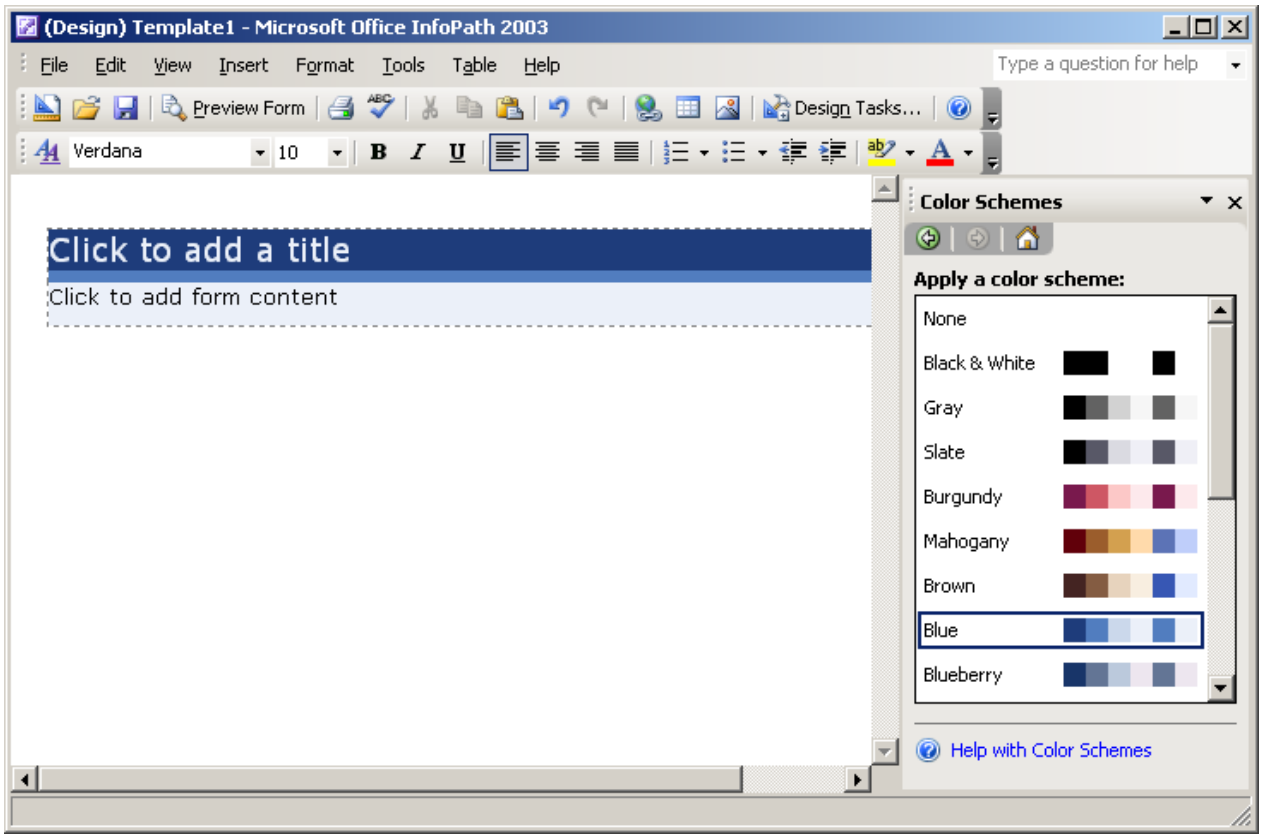


Il s'agit bien de la structure de notre XSD.

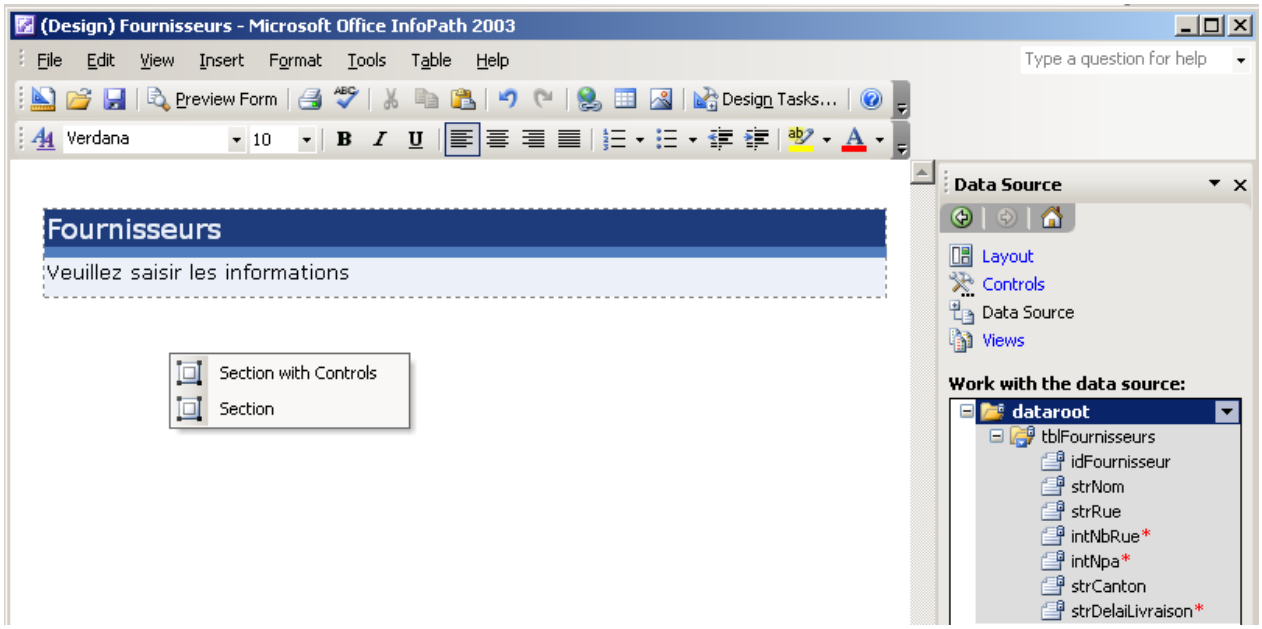
Ensuite dans le layout du formulaire choisissez *Table with Title:*



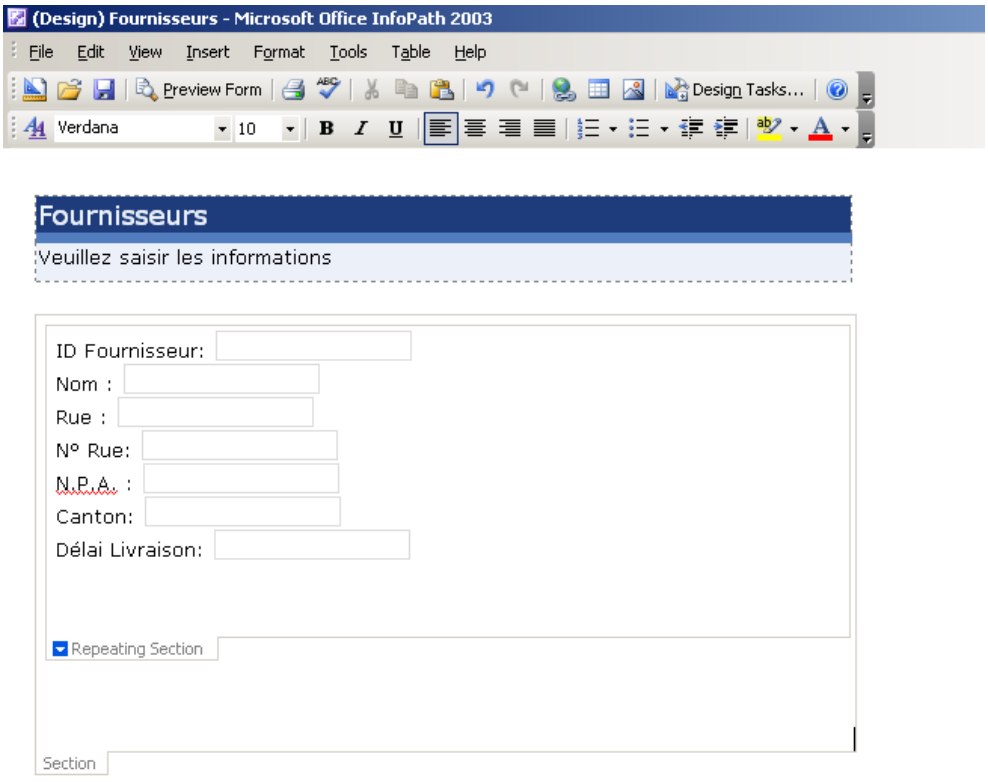
et ensuite, dans choisissons un joli design:



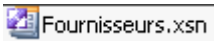
Ensuite retournez dans *Data Source* et faites un glisser-déplacer du nœud *dataroot* dans la feuille en sélectionnant *Section with controls*:



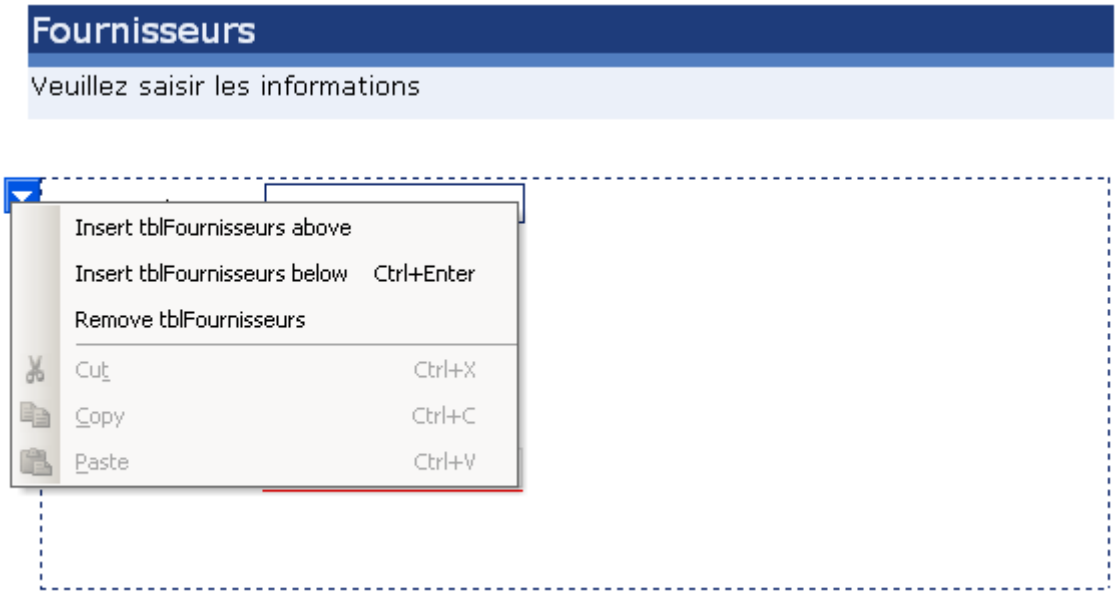
et voilà la résultat (après avoir un peu changé les texte des libellés):



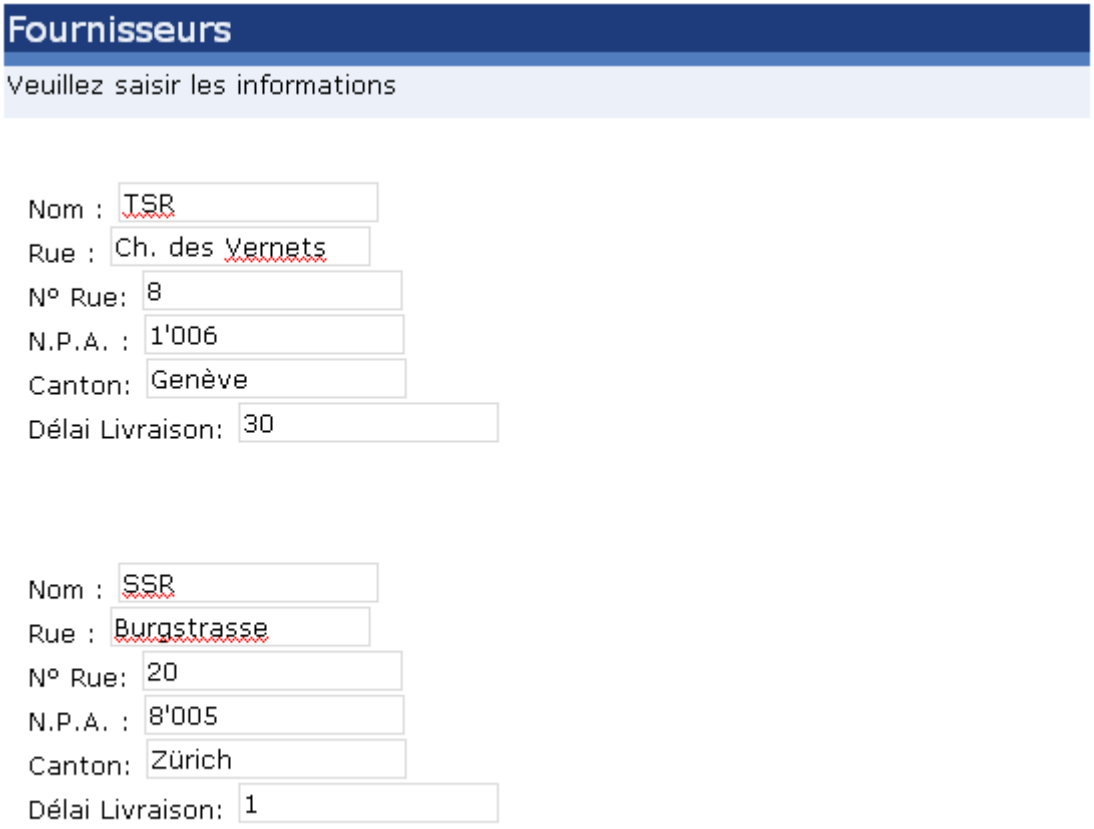
Enregistrez les modifications et fermez InfoPath. Faites ensuite un double clique sur le fichier InfoPath:



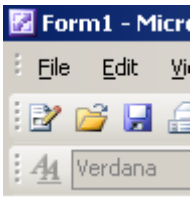
et saisissez les données. Au besoin, en faisant un clic sur la flèche bleue, vous pouvez ajouter des nouveaux enregistrements (c'est ce qui fait d'InfoPath un outil très puissant):



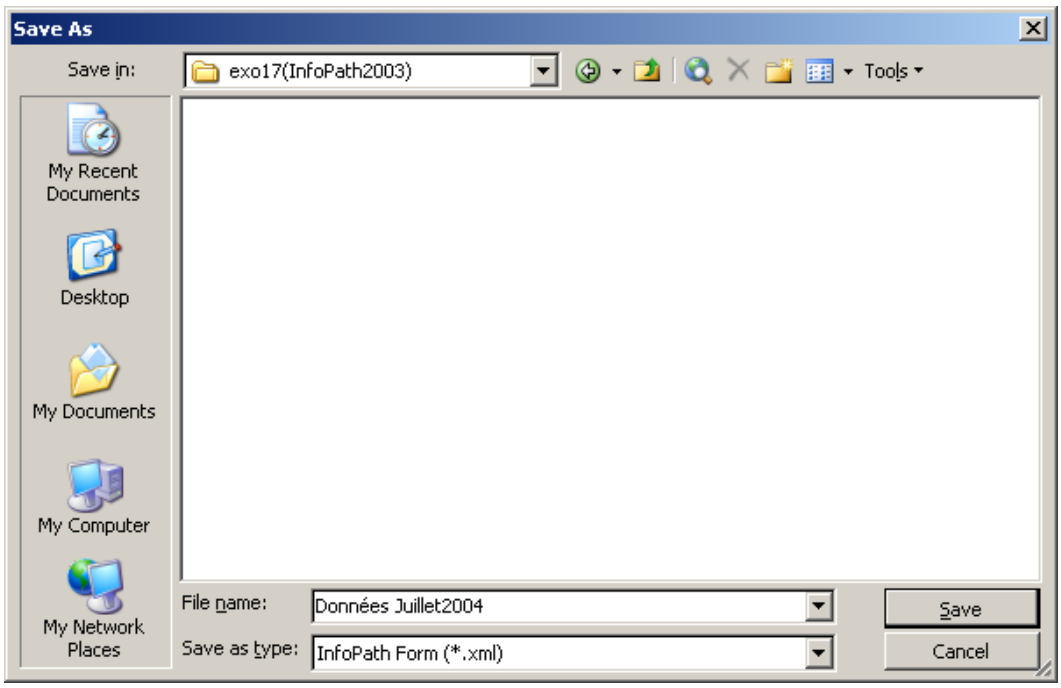
et donc après avoir saisi quelques valeurs:



L'utilisateur du formulaire clique sur *Enregistrer:*



et InfoPath propose automatiquement de sauvegarder les données en XML:



et voici le fichier XML sortant:

```
<?xml version="1.0" encoding="UTF-8"?>
<?mso-infoPathSolution solutionVersion="1.0.0.3" productVersion="11.0.5531" PIVersion="1.0.0.0"
href="file:///C:\Documents%20and%20Settings\Isoz\Mes%20documents\Professionel\Cours\XML\exo17(InfoPath2003)\
Fournisseurs.xsn" language="fr" ?>
<?mso-application progid="InfoPath.Document"?>
<dataroot>
    <tblFournisseurs>
        <idFournisseur/>
        <strNom>TSR</strNom>
        <strRue>Ch. des vernets</strRue>
        <intNbRue>8</intNbRue>
        <intNpa>1006</intNpa>
        <strCanton>Genève</strCanton>
        <strDelaiLivraison>30</strDelaiLivraison>
    </tblFournisseurs>
    <tblFournisseurs>
        <idFournisseur/>
        <strNom>SSR</strNom>
        <strRue>Burgstrasse</strRue>
        <intNbRue>20</intNbRue>
        <intNpa>8005</intNpa>
        <strCanton>Zürich</strCanton>
        <strDelaiLivraison>1</strDelaiLivraison>
    </tblFournisseurs>
</dataroot>
```

Tout est parfait (mis à part la ligne propriétaire de Microsoft…) pour un traitement par MS Excel 2003, pour un formatage web par un XSL, pour une utilisation de pagination dans MS Word 2003 ou pour un import dans MS Access 2003. Rien de nouveau donc de ce côté.

L'employé n'a qu'à envoyer ce document XML par e-mail à son responsable ou à des collègues.

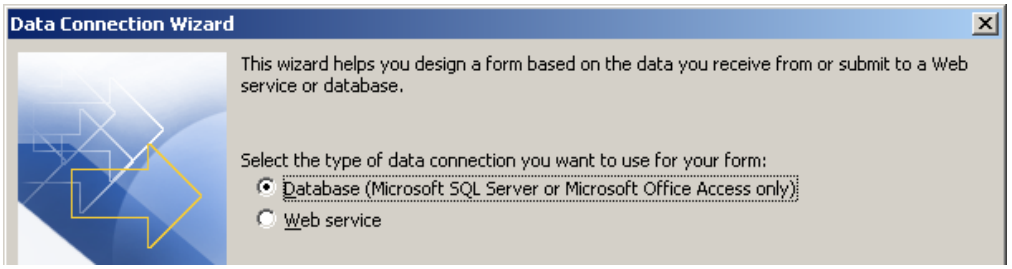
10.4.2 Liaison dynamique (exemple 2)

Nous allons maintenant faire un formulaire dynamique lié directement à la table *tblFournisseurs* de notre base MS Access avec des listes déroulantes, etc. etc. L'échange des données entre MS Access et MS InfoPath se faisant par ADO (donc sans faire usage du XML mais bon c'est quand même bien de faire un exemple ne serait-ce que pour voir la facilité d'usage de MS Infopath).

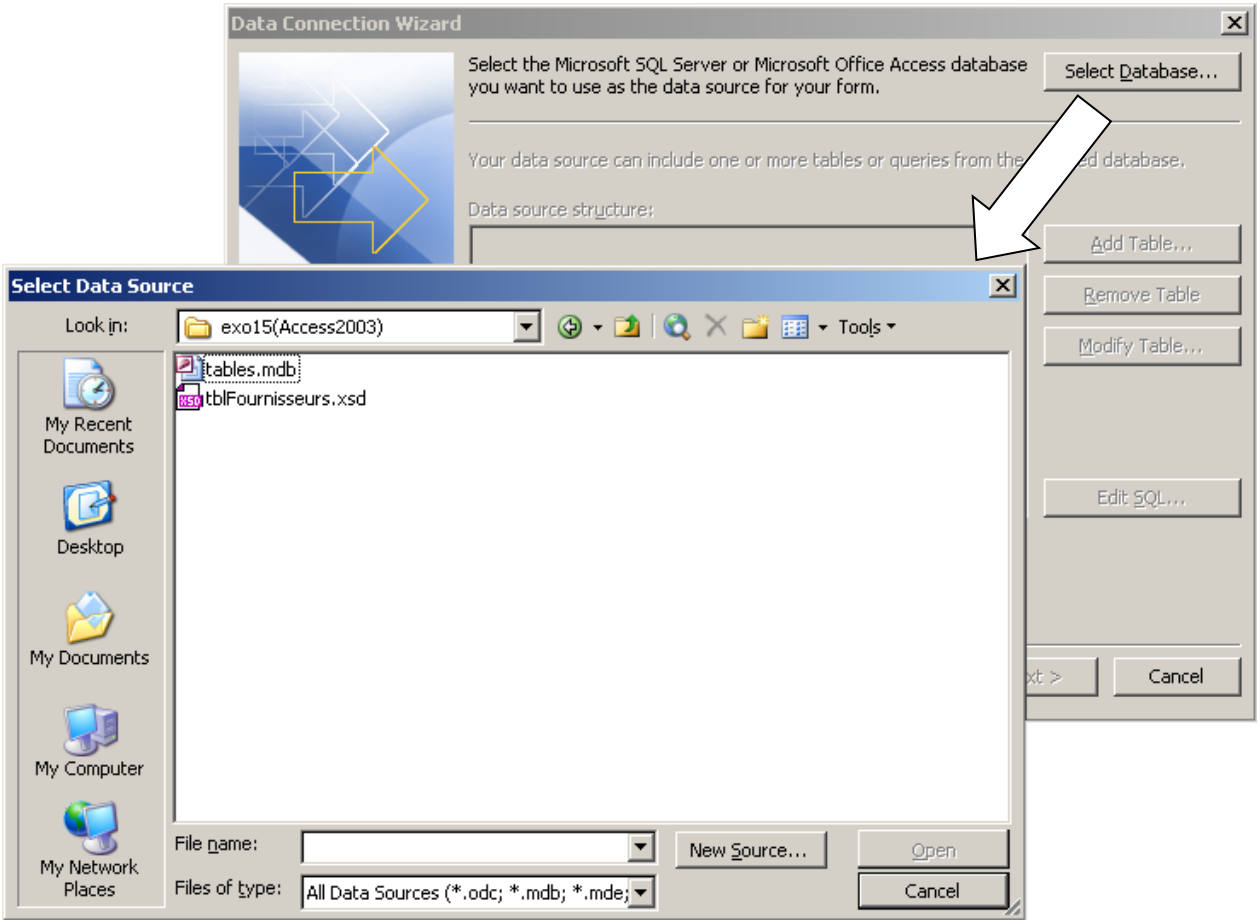
Ouvrez InfoPath (SP1) et sélectionnez:

Dans le volet Office cliquez sur:

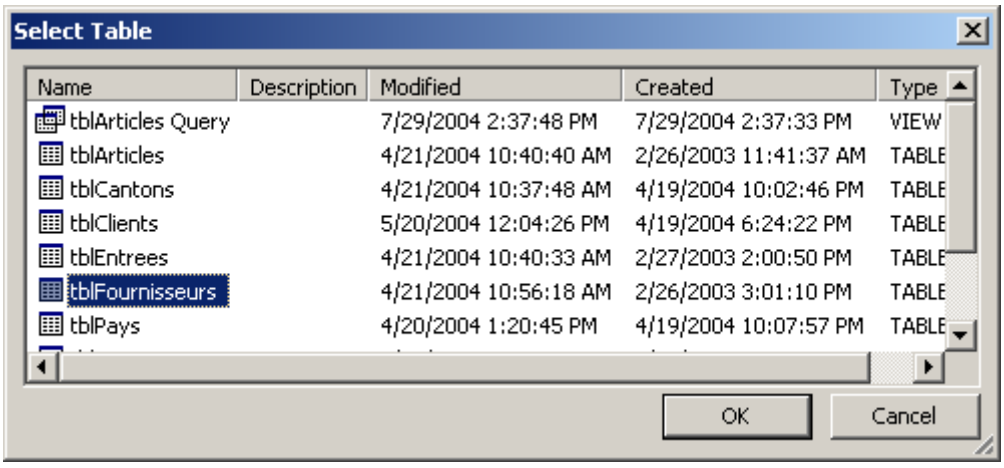
Afin que nous puissions nous connecter à la base MS Access (les Webservices utilisant les flux XML par SOAP ne seront pas traités dans ce document – c'est que pour les spécialistes):



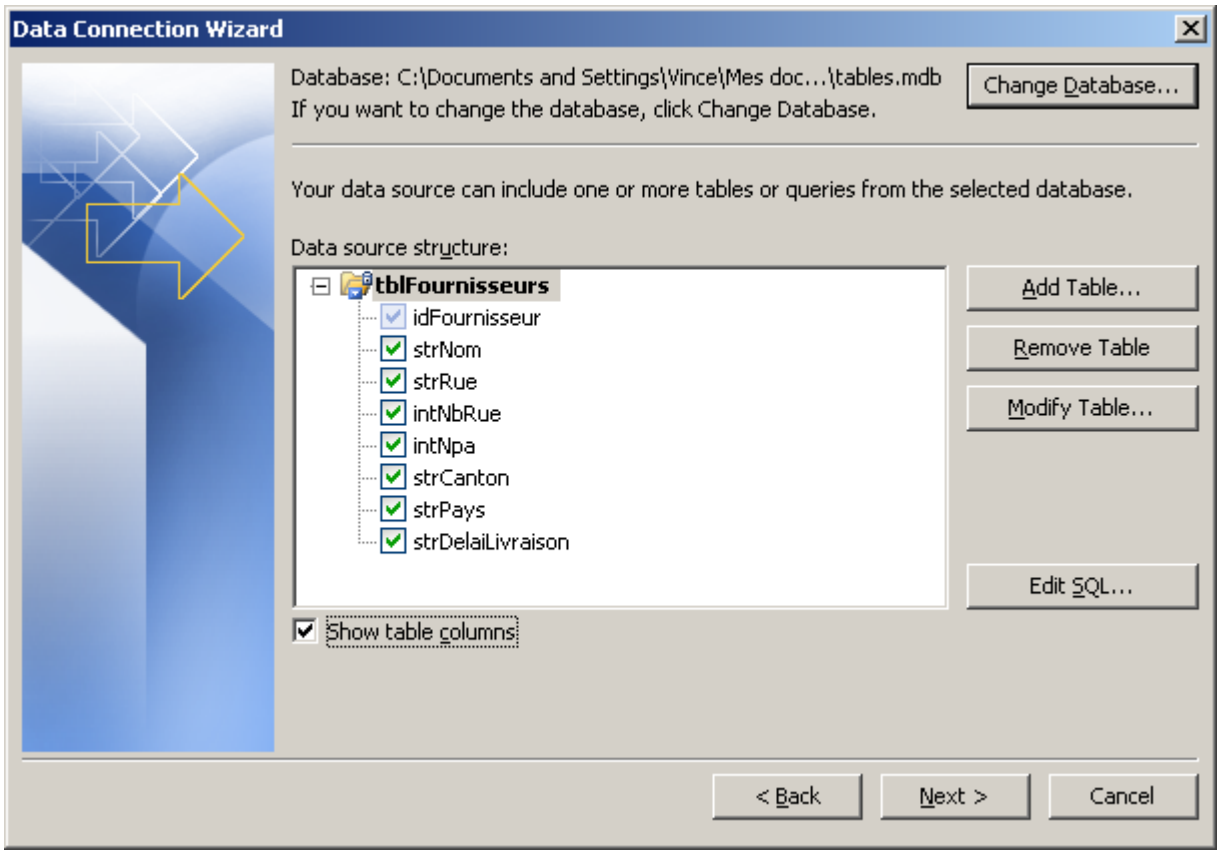
Ensuite la fenêtre suivante apparaît (cliquez sur *Next*):



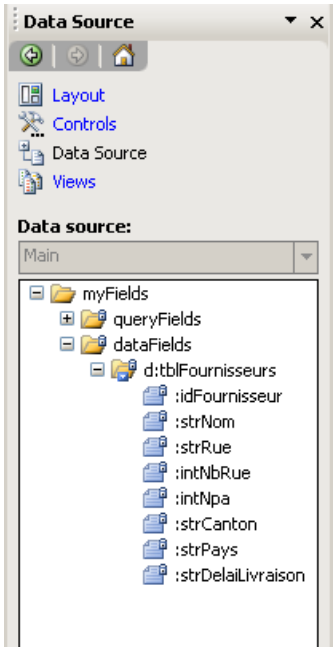
et après avoir sélectionné *tables.mdb* sélectionnez *tblFournisseurs:*



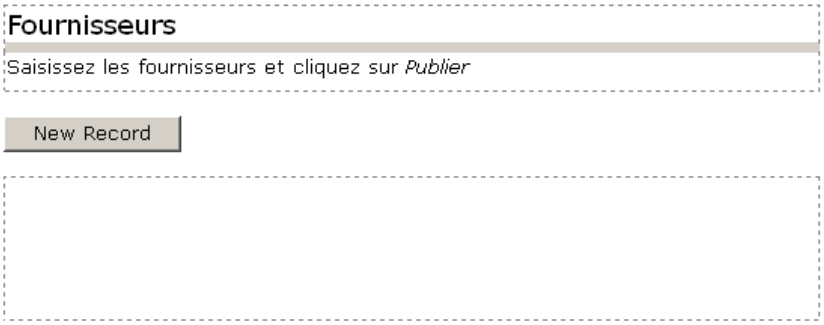
et vous devez avoir:



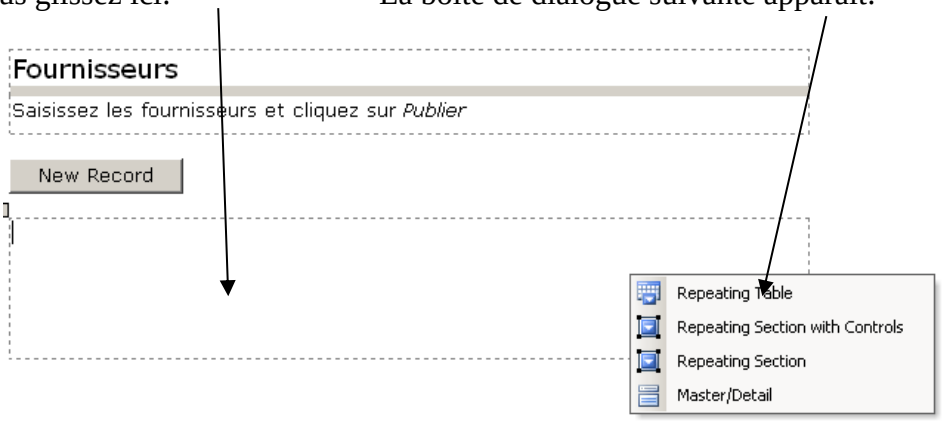
Cliquez sur *Next*, donnez un nom à votre connexion (*Fournisseurs* par exemple) et sur *Finish*. Vous devrez avoir le résultat suivant dans le volet Office:



A gauche sur la feuille (InfoPath SP1), vous avez une zone *Drag data fields here* (vous pouvez effacer la partie concernant la *Query*). Glissez-y le nœud *d:tblFournisseurs:*

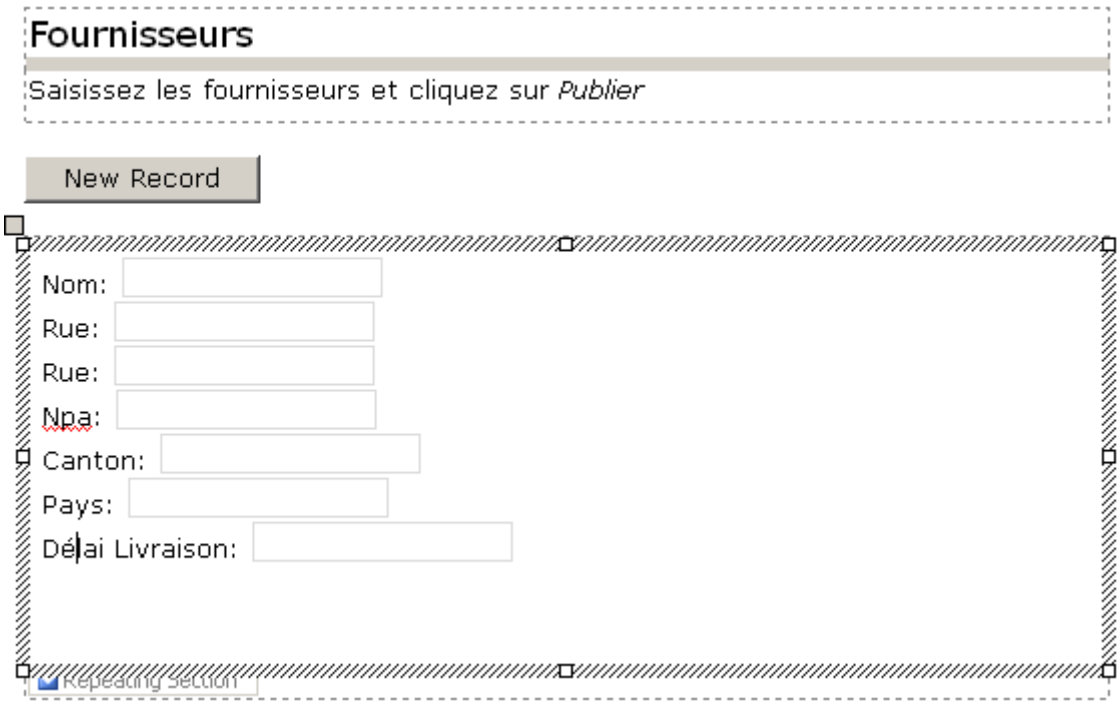


Quand vous glissez ici: La boîte de dialogue suivante apparaît:



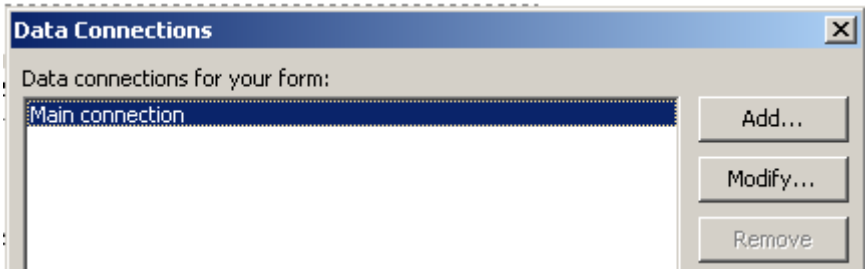
Dans la boîte de dialogue, sélectionnez *Repeating Section with Controls.* Vous aurez:

Vous pouvez changer le nom des étiquettes.et supprimer le champ *id Fournisseur* (puisque c'est une clé primaire *AutoNumber* dans notre base Access):

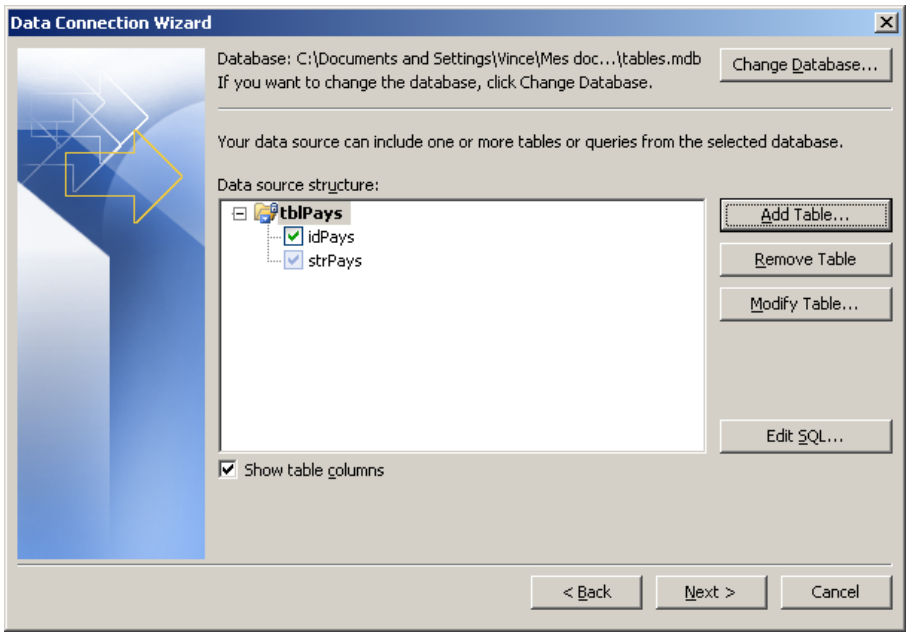


Maintenant nous aimerions que lorsque l'utilisateur clique sur *Canton* ou *Pays* que la liste des cantons ou pays disponibles dans la base Access (dans les tables *tblCantons* et *tblPays*) s'affiche à l'écran (pour que l'utilisateur n'ait pas à les saisir).

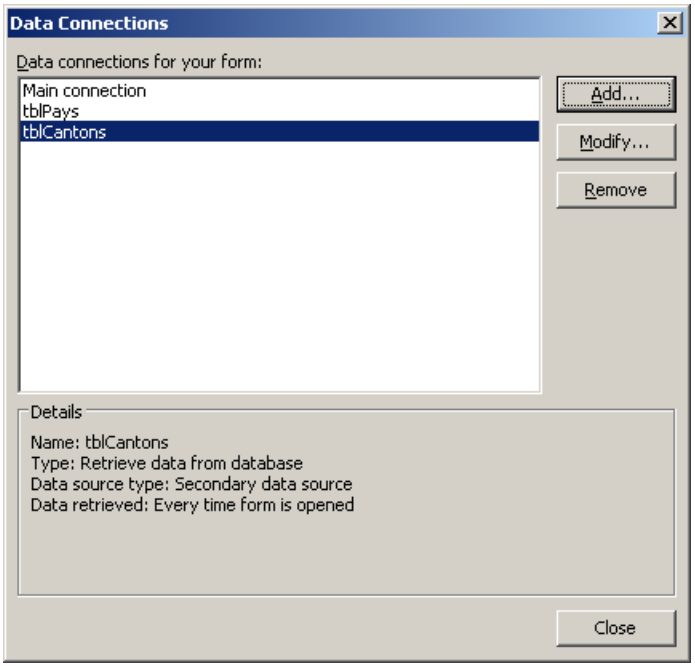
Pour ce faire, il faut aller dans le menu *Tools/Data Connection*:



*Add…*ensuite *Receive Data* ensuite *Database (Microsoft SQL Server or Microsoft Access only)* ensuite *Select Database* et allez chercher la base *tables.mdb* et la table *tblPays*:



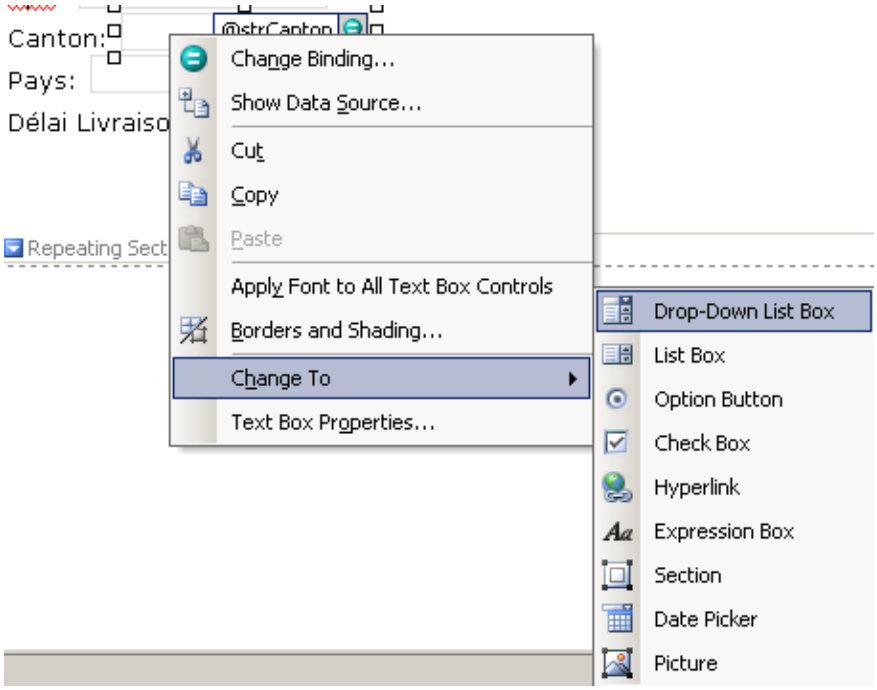
Cliquez sur *Next*, donnez un nom à la connexion et recommencez l'opération pour la table des cantons. Vous aurez alors dans la fenêtre des connexions (*Tools/Data connections*) trois connexions:



Cliquez sur *Close*.

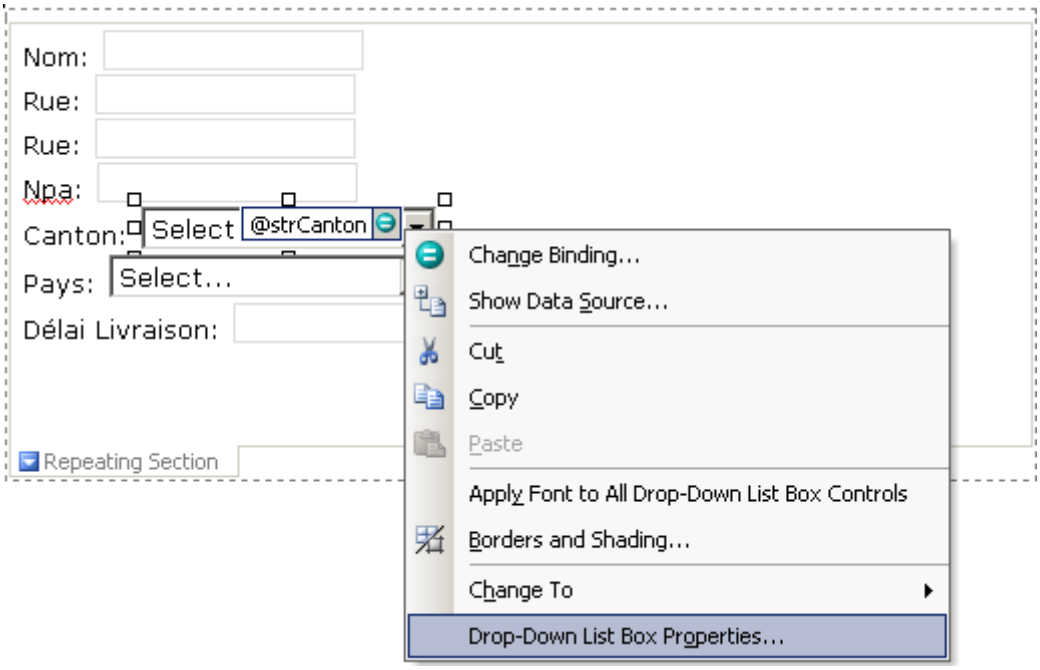
Pour relier les champs *Cantons* et *Pays* du formulaire InfoPath aux tables de la base Access et les transformer en listbox voici comment il faut procéder:

Il faut d'abord transformer les champs texte en Drop Down Listbox. Pour ce faire, bouton droit et:

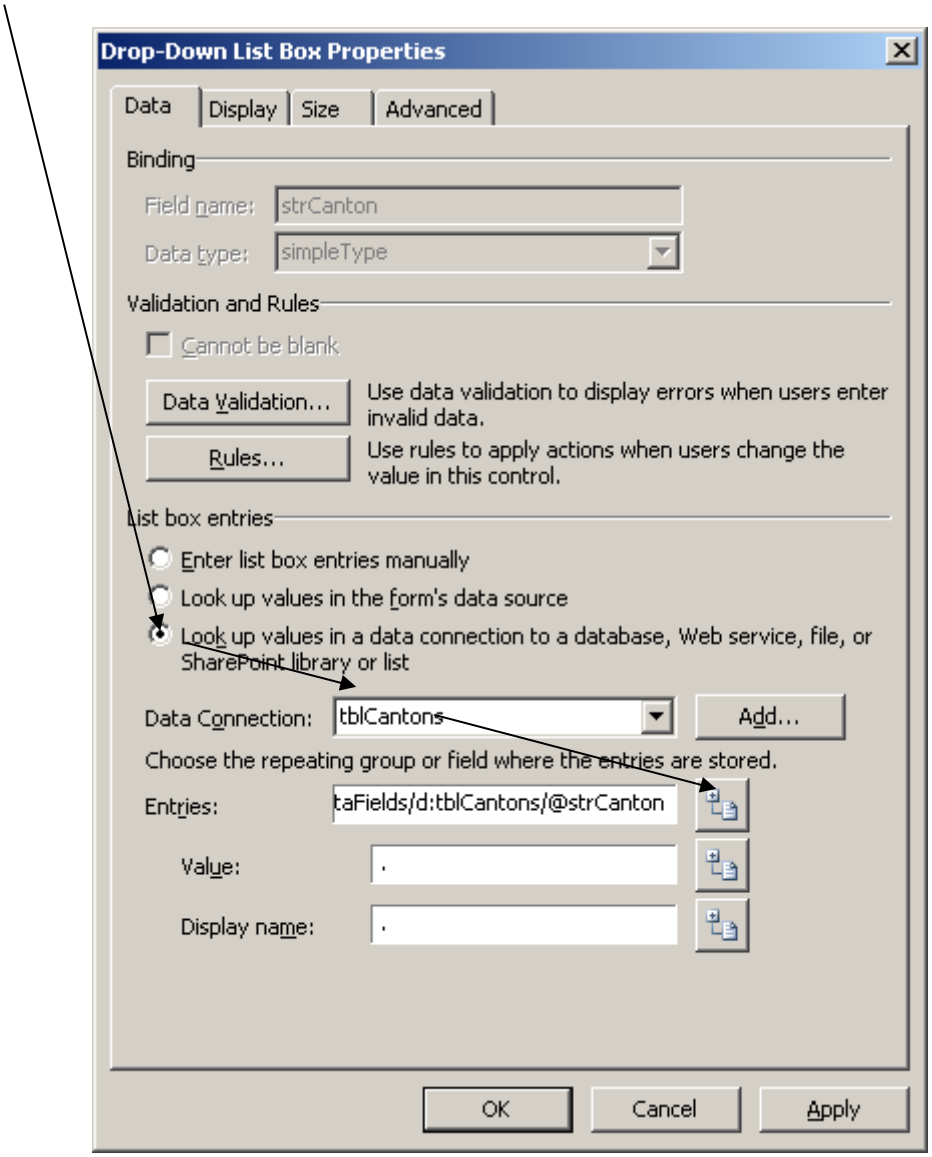


et idem pour les pays.

Maintenant il faut "peupler" les ListBox avec le contenu des tables de la base Access. Pour ce, clique droit et:

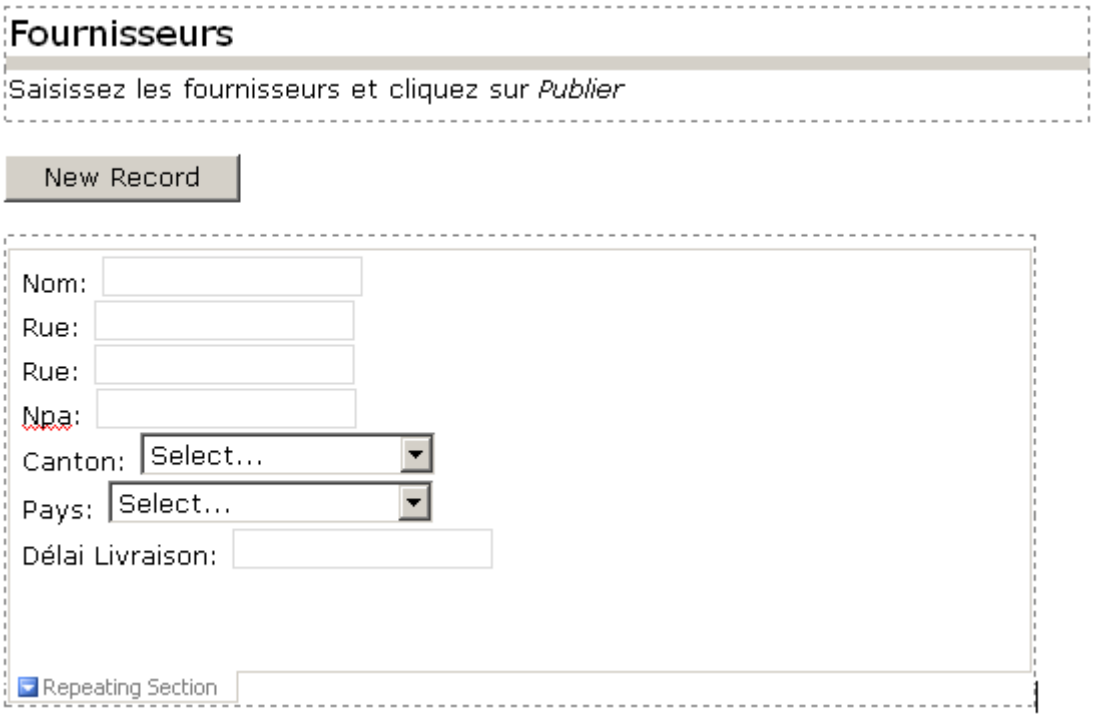


Ensuite:

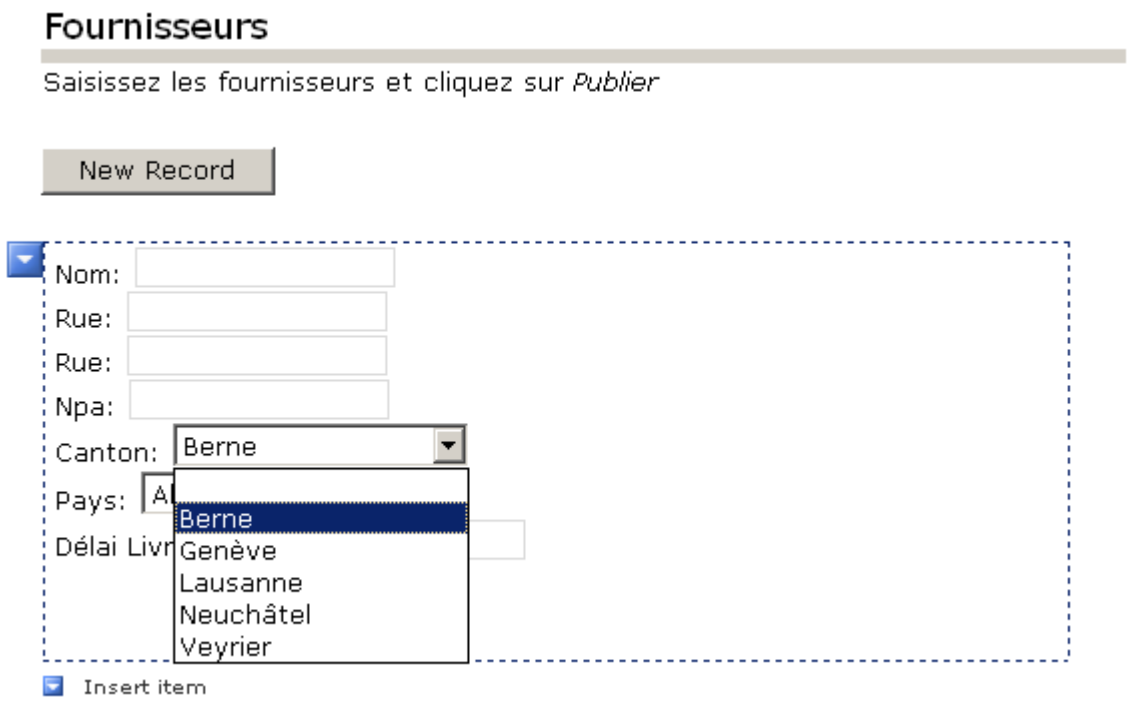


Vous cliquez sur *OK* et vous faites idem pour les pays.

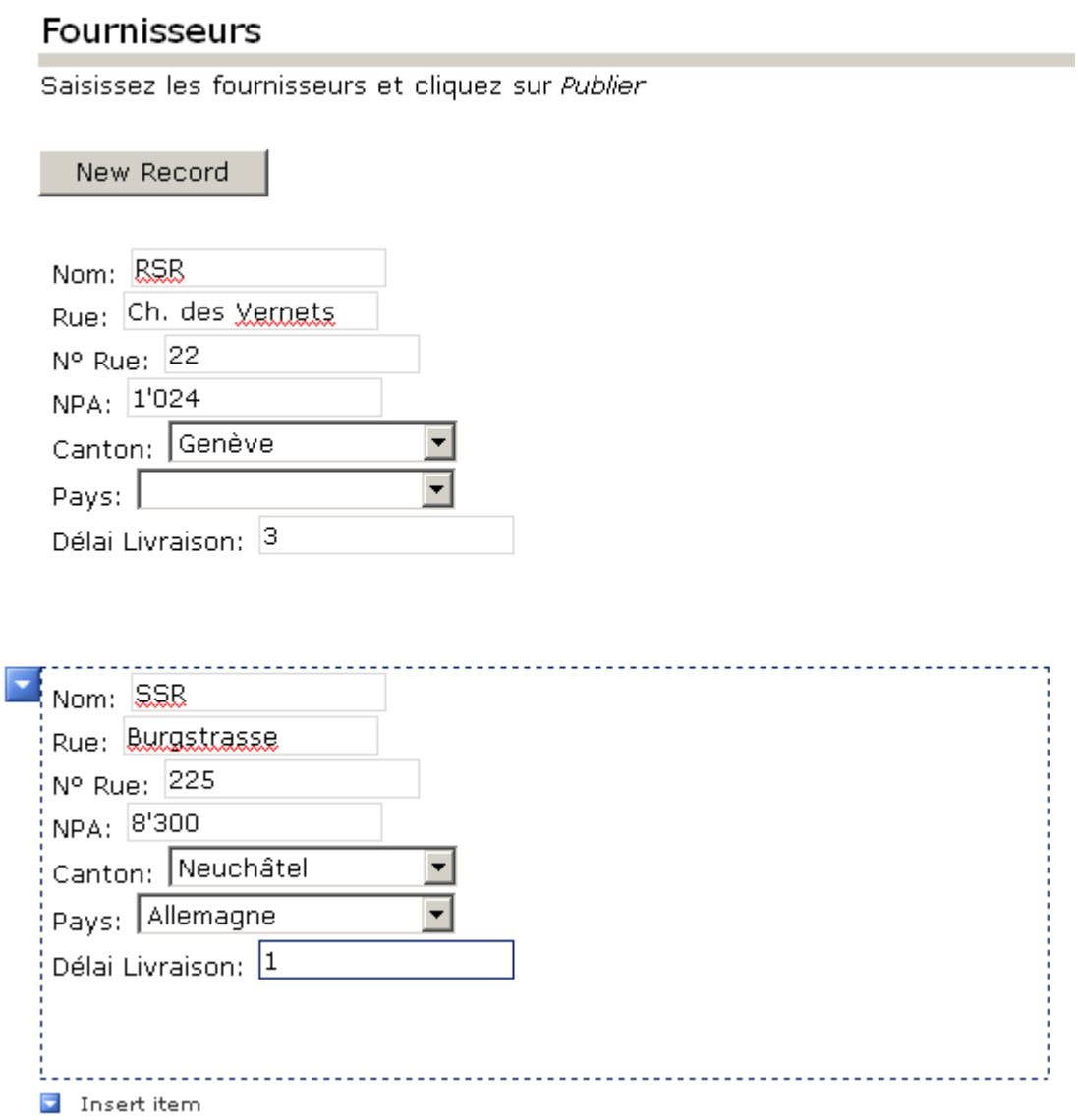
Vous enregistrez votre formulaire (*.xsn) et l'ouvrez par un double clique depuis votre raccourci sur le bureau ou votre explorateur Windows (si lorsque vous l'ouvrez il vous dit que les données se trouve sur un autre domaine vous validez par *Oui*):



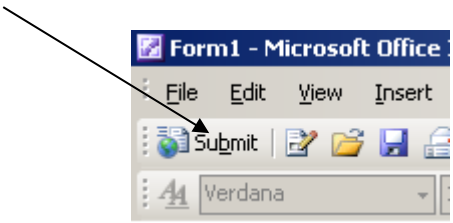
Cliquez sur une des ListBox et oh miracle:



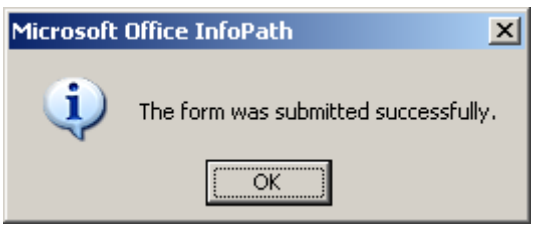
et maintenant pour la saisie de données ? Eh ben c'est très simple, on saisit le premier *Record* et le deuxième (cliquez sur *Insert Item*) etc…:



et quand c'est fini, on clique sur:

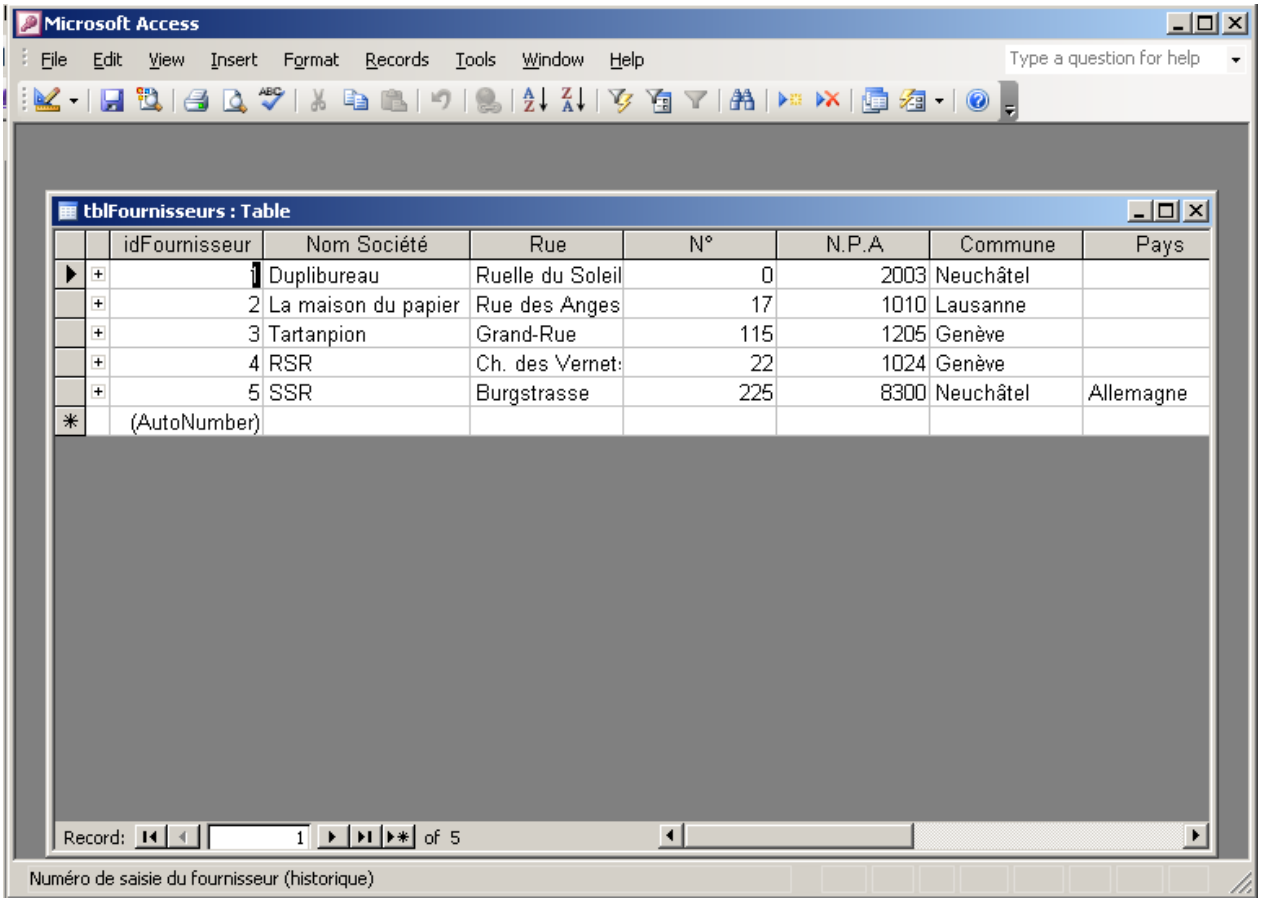


on a:



☺ on est forts hein…

Et si nous allons ouvrir notre table Access *tblFournisseurs*, nous avons bien:



Remarque: pour supprimer le bouton d'édition de MS InfoPath sur les postes clients allez dans *Tools/Options* et activez *Enable protection*.

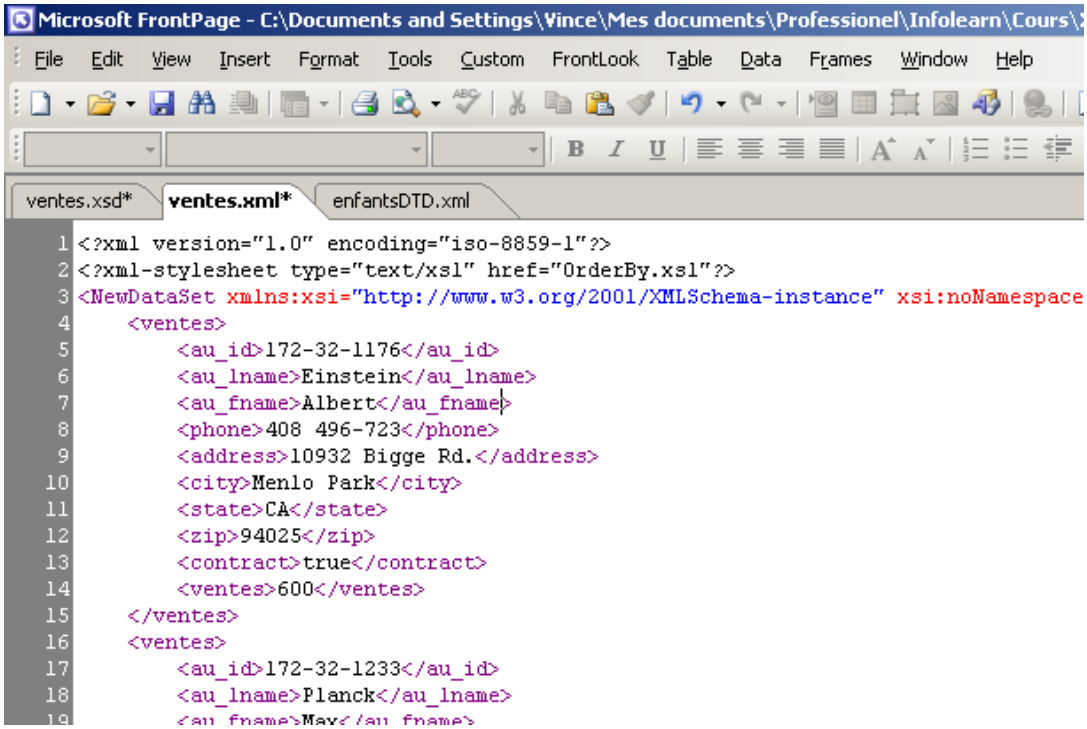
Voilà pour les bases de MS InfoPath et XML.

Bien sûr il faut avoir en tête que tout cela communique avec l'ensemble des produits de la gamme MS Office 2003.

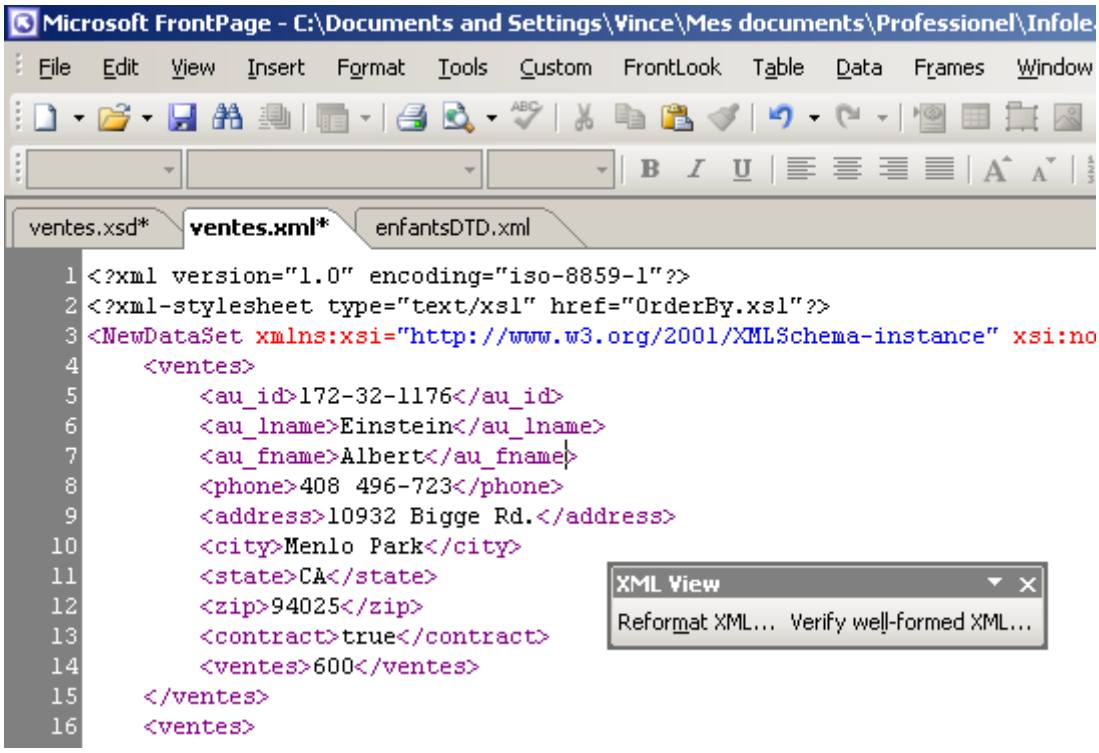
10.5 MS Frontpage 2003

Oui il faut en parler un petit peu… MS FrontPage 2003 contient quelques nouveautés (deux) par rapport aux fichiers XML, XSL, XSD, etc…

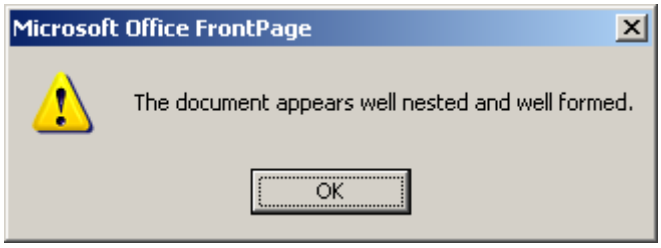
Première nouveauté: Vous pouvez créer des fichiers XML, XSL, XSD, DTD dans FrontPage 2003 et ceux-ci sont colorisés comme il se doit:



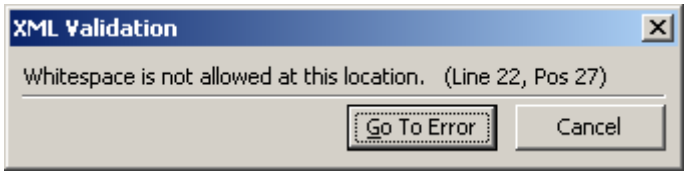
Deuxième nouveauté: il y a une barre d'outils qui permet de mettre en forme le XML (au cas où vous l'auriez mal fait… ce qui est dur…) et de vérifier s'il est valide:



Document valide:



Document non valide:



Bof mis à part ça concernant le XML j'ai rien vu d'autre (au fait si mais… bon je suis pas un fan de frontpage mis à part quand il s'agit de faire du SharePoint ou des équations…).

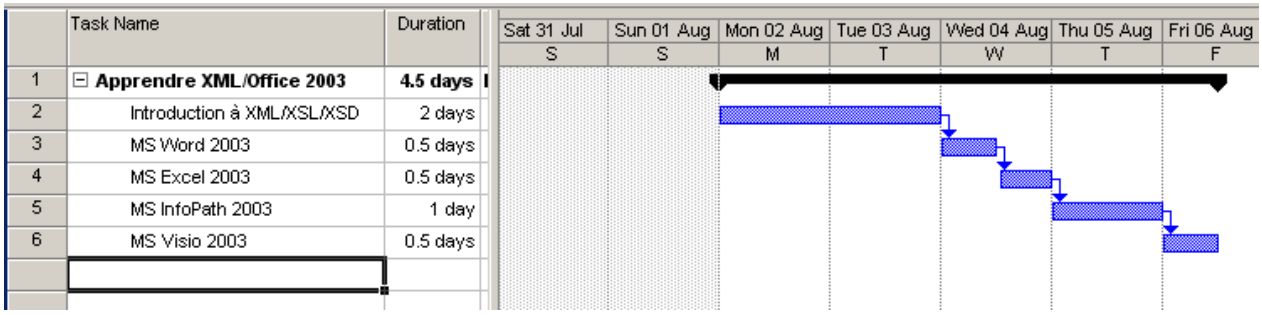
10.6 MS Project 2003

A nouveau, MS Project 2003 permet lui aussi d'exporter ou d'importer des fichiers XML.

Cette possibilité permet ainsi à des développeurs de créer des logiciels qui génèrent du XML compatible avec MS Project 2003 (pour en faire des diagrammes de Gantt ou de Pert) et ceci sans passer nécessairement par la manne Microsoft (c'est aussi un des nombreux intérêts du XML). Seule exigence, connaître le schéma XSD propriétaire de MS Project.

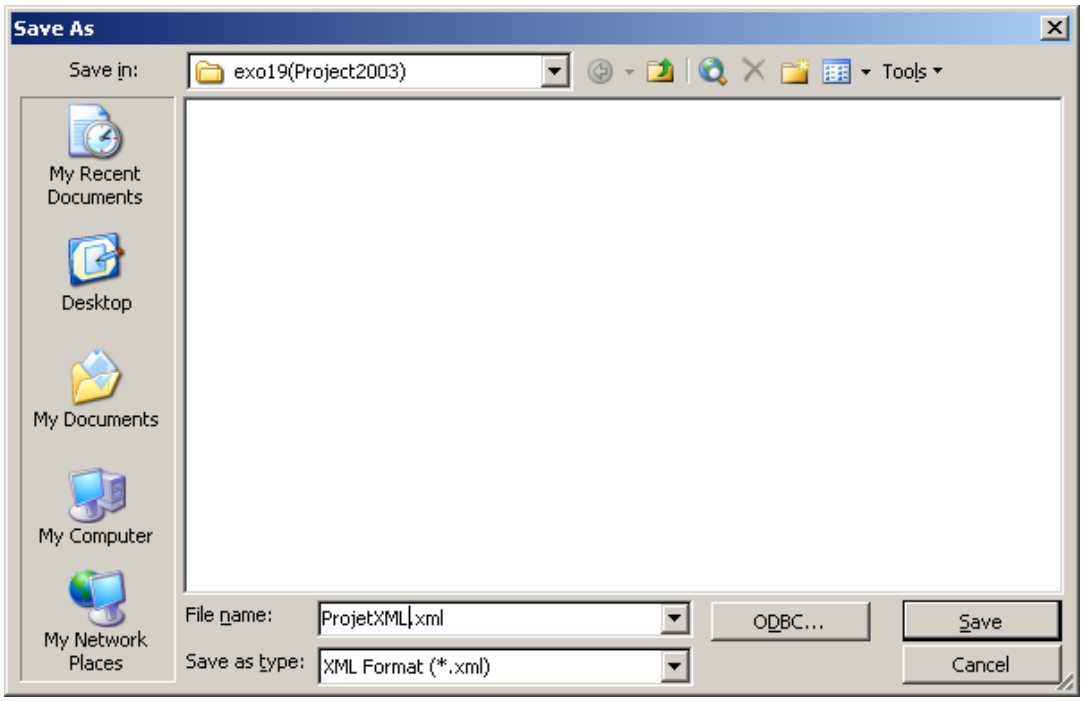
Petit problème cependant (que nous retrouverons dans MS Visio 2003), le fichier XML résultant d'une exportation est très lourd au niveau du nombre de balises. Un traitement par voie de programmation est quasiment indispensable.

Voici notre projet de base pour l'exemple (toujours le même: apprendre XML ☺):



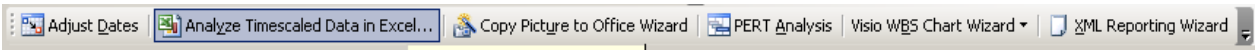
Remarques: nous n'avons spécifié aucune ressource ni quoi que ce soit de particulier dans le calendrier.

Allez dans *Fichier/Enregistrer sous* et choisissez comme type de fichier *XML*:

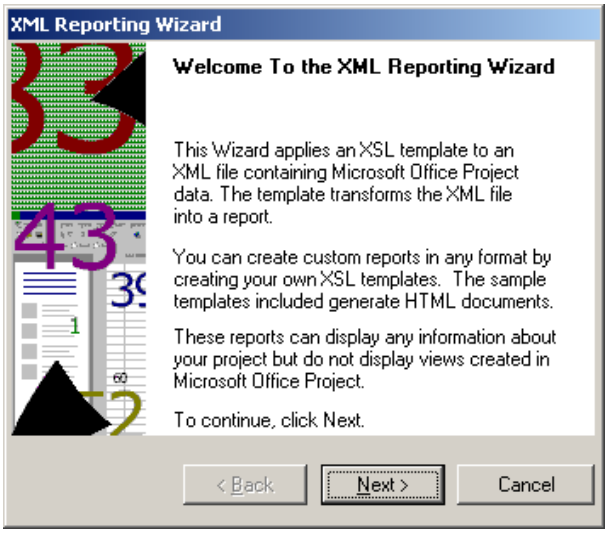


Le fichier XML résultant est tellement conséquent (c'est normal ceci dit…) qu'il serait indécent de gaspiller des pages électroniques pour le présenter. C'est le travail du développeur de le décortiquer pour créer un parseur adapté.

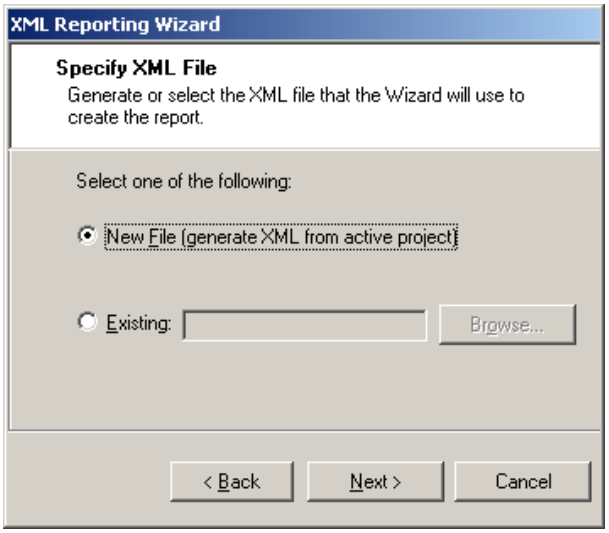
Cependant !!!! Il existe un nouveau bouton dans MS Project 2003 fait pour générer des rapports XML d'une tout beauté. Toujours sur le même exemple de base, activons la barre d'outils *Analysis*:

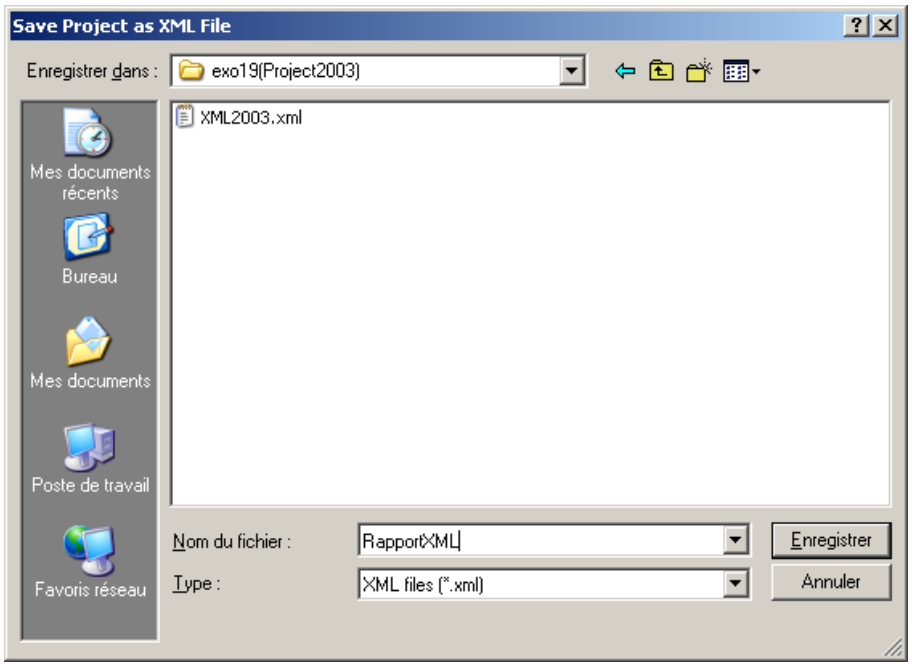


et observez bien le dernier bouton tout à droite *XML Reporting Wizard* ;-P. Et bien cliquons dessus:

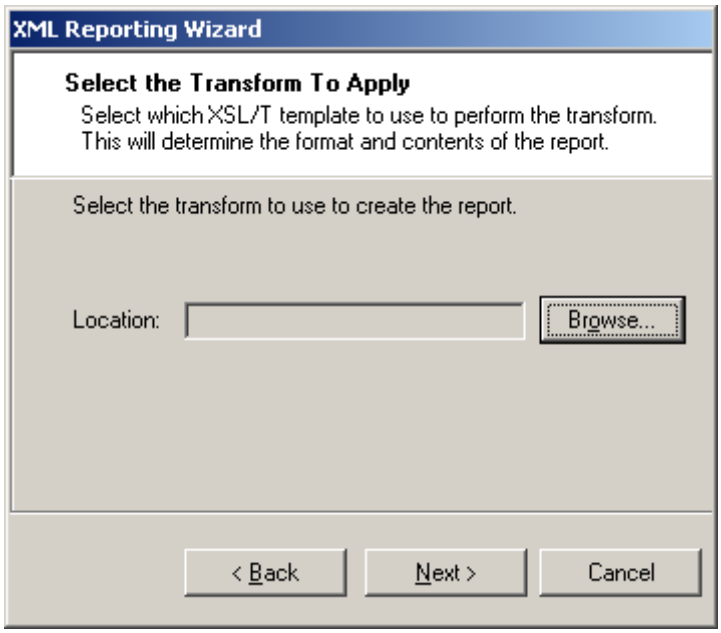


Deuxième étape, où il faudra dire où est-ce que vous souhaitez enregistrer le fichier XML résultant:

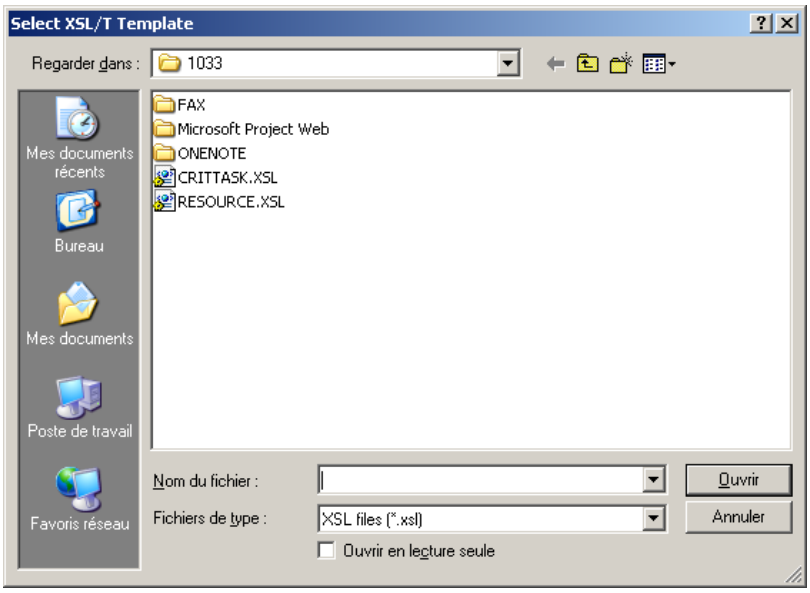




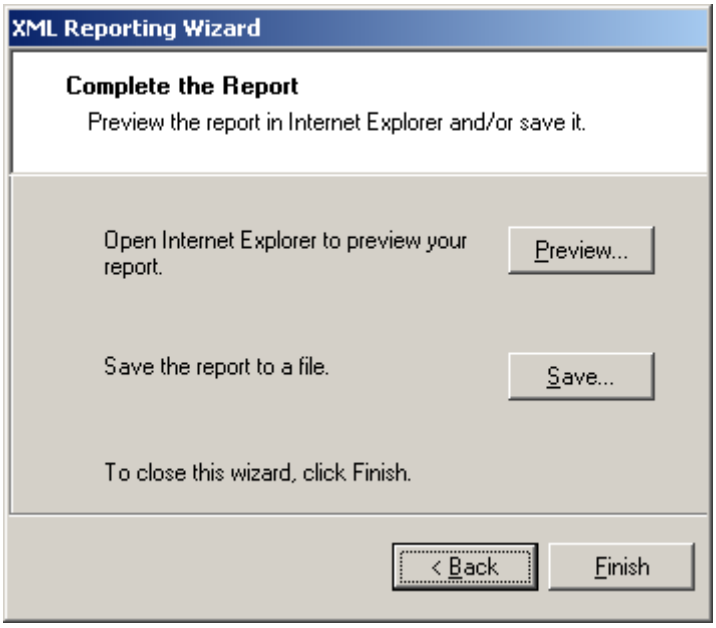
Une fois ceci fait, il vous demande le fichier de formatage XSL (par défaut il en existe deux dans le dossier d'installation de MS Project):



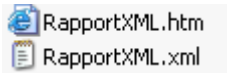
Voici les deux fichiers en question:



Prenons les tâches critiques:



On enregistre le rapport sous un fichier HTML et:



Voilà à quoi cela ressemble:

1. Le fichier XML:

….trop long… et toujours indigeste … ;-)

2. Le fichier XSL (qui lui est beaucoup plus digeste):

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

```
   <HTML>
    <BODY BGCOLOR="white">
     <TABLE BORDER="0" CELLSPACING="1">
      <TR>
       <TD COLSPAN="5" ALIGN="center">
        <H2><font face="tahoma" size="5">Critical Tasks for Project: <xsl:value-of select="Project/Name" />
</font></H2>
       </TD>
      </TR>
      <TR BGCOLOR="0x333FF">
        <TH><font color="white">ID</font></TH>
        <TH><font color="white">Name</font></TH>
        <TH><font color="white">Priority</font></TH>
        <TH><font color="white">Start</font></TH>
        <TH><font color="white">Finish</font></TH>
       </TR>
       <xsl:for-each select="Project/Tasks/Task">
        <xsl:if test="Critical[.=1]">
        <xsl:if test="Summary[.=0]">
          <TR>
            <TD><xsl:value-of select="ID"/></TD>
            <TD><xsl:value-of select="Name"/></TD>
            <TD><xsl:value-of select="Priority"/></TD>
            <TD><xsl:value-of select="Start"/></TD>
            <TD><xsl:value-of select="Finish"/></TD>
          </TR>
        </xsl:if>
        </xsl:if>
       </xsl:for-each>
     </TABLE>
    </BODY>
   </HTML>
  </xsl:template>
</xsl:stylesheet>
```

3. Le fichier HTML (le combiné du fichier XSL et XML donc…):

Critical Tasks for Project: RapportXML.xml

ID	Name	Priority	Start	Finish
2	Introduction à XML/XSL/XSD	500	2004-08-02T08:00:00	2004-08-03T17:00:00
3	MS Word 2003	500	2004-08-04T08:00:00	2004-08-04T12:00:00
4	MS Excel 2003	500	2004-08-04T13:00:00	2004-08-04T17:00:00
5	MS InfoPath 2003	500	2004-08-05T08:00:00	2004-08-05T17:00:00
6	MS Visio 2003	500	2004-08-06T08:00:00	2004-08-06T12:00:00

Mais franchement… que demande le peuple !? C'est pas beau ça ?

10.7 MS Visio 2003

A nouveau, MS Visio 2003 permet lui aussi d'exporter ou d'importer de nombreux type des schémas (Brainstorming, Business Process Report, etc.) au format XML.

Cette possibilité permet ainsi à des développeurs de créer des logiciels qui génèrent du XML compatible avec MS Visio (pour en faire des schémas) et ceci sans passer par la manne Microsoft (c'est aussi un des nombreux intérêts du XML).

Remarque: curieusement Microsoft à créé un outil d'export XML de schémas Brainstorming automatique mais par pour les Organisation Chart ou les Business Process qui sont quand même nettement plus utilisés (effet de mode des schémas Brainstorming certainement…).

Nous allons donner donc deux exemples:

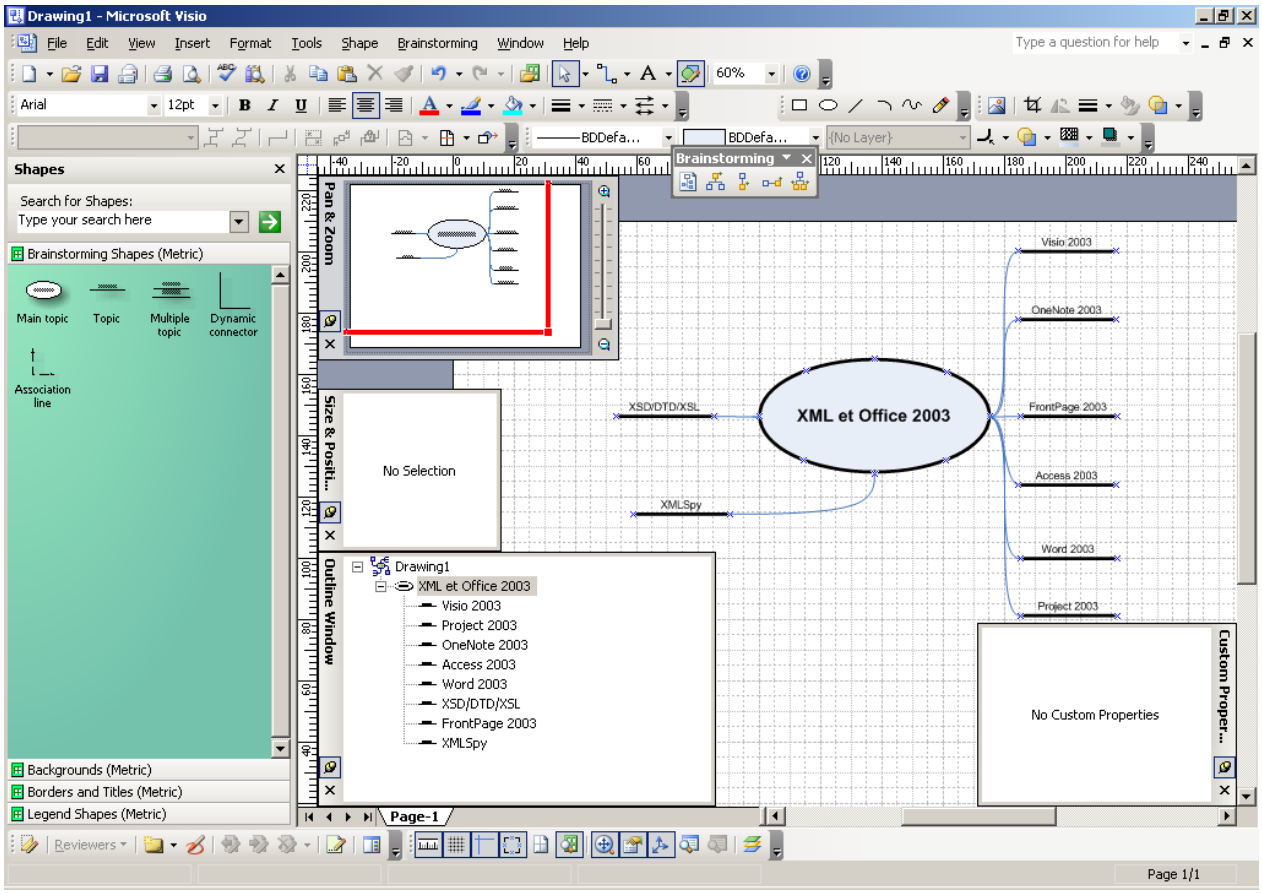
1. Celui d'un Export/Import de schéma Brainstorming (sic!)

2. Celui d'un Export d'un rapport d'un Business Process (ça j'adore !)

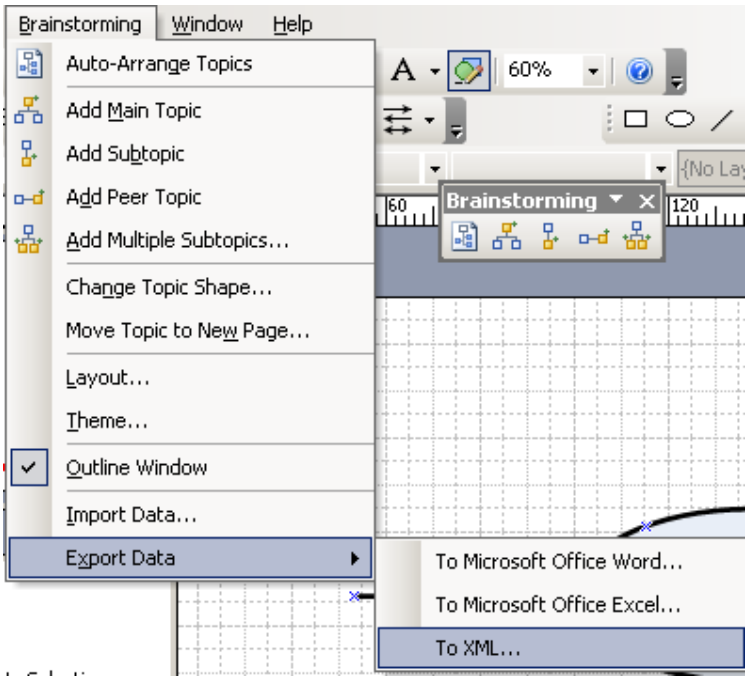
10.7.1 Génération diagramme (exemple 1)

Comme d'habitude le but de ce document n'est pas d'apprendre à utiliser MS Visio donc je supposerai que vous savez créer des schémas de Brainstorming (ce qui est du niveau d'un enfant disons… de 12 ans).

Donc voilà un diagramme tout fait sur un sujet que vous connaissez bien (il faut avoir des idées parfois…):



Pour l'exporter en XML (afin de voir comment cela est fait et développer un logiciel qui en crée un semblable pour l'import ce coup-ci):



Ce qui donne ma foi un schéma extrêmement simple à reproduire à l'aide de code VBA, C#, VB.Net ou Java, etc.:

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

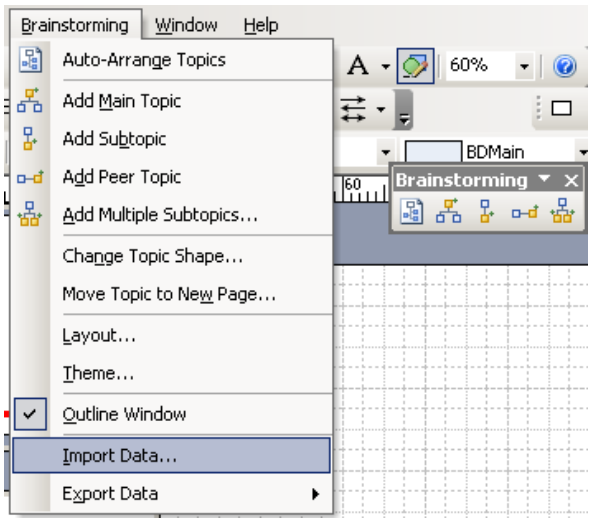
```
<bs:Brainstorm xmlns:bs="http://schemas.microsoft.com/visio/2003/brainstorming">
    <bs:topic bs:TopicID="T1">
        <bs:text>XML et Office 2003</bs:text>
        <bs:topic bs:TopicID="T1.1">
            <bs:text>Visio 2003</bs:text>
        </bs:topic>
        <bs:topic bs:TopicID="T1.2">
            <bs:text>Project 2003</bs:text>
        </bs:topic>
        <bs:topic bs:TopicID="T1.3">
            <bs:text>OneNote 2003</bs:text>
        </bs:topic>
        <bs:topic bs:TopicID="T1.4">
            <bs:text>Access 2003</bs:text>
        </bs:topic>
        <bs:topic bs:TopicID="T1.5">
            <bs:text>Word 2003</bs:text>
        </bs:topic>
        <bs:topic bs:TopicID="T1.6">
            <bs:text>XSD/DTD/XSL</bs:text>
        </bs:topic>
        <bs:topic bs:TopicID="T1.7">
            <bs:text>FrontPage 2003</bs:text>
        </bs:topic>
        <bs:topic bs:TopicID="T1.8">
            <bs:text>XMLSpy</bs:text>
        </bs:topic>
    </bs:topic>
</bs:Brainstorm>
```

A nouveau, on peut très bien imaginer envoyer ces données dans une base MS Access ou dans Excel, ou dans Word que ce soit en import ou/et en export.

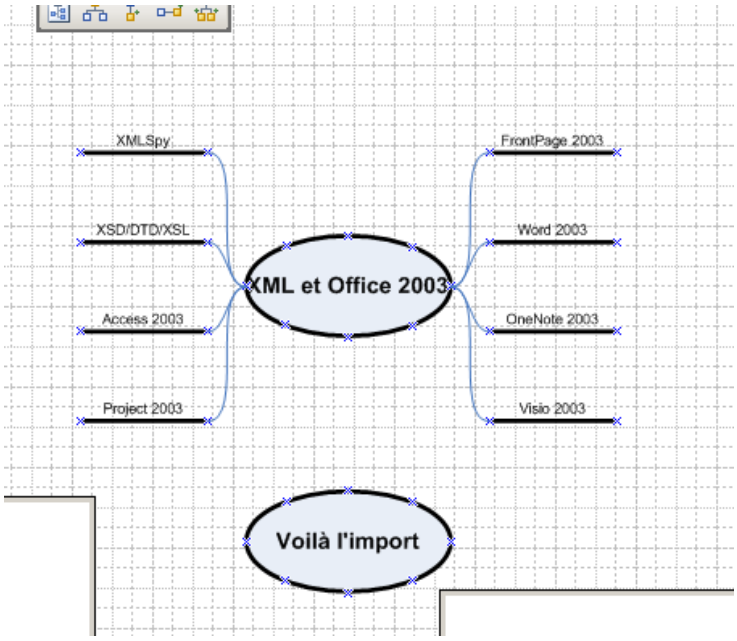
Ajoutons un noeud T2 "*Voilà l'import*" manuellement:

```
<?xml version="1.0" encoding="UTF-8"?>
<bs:Brainstorm xmlns:bs="http://schemas.microsoft.com/visio/2003/brainstorming">
    <bs:topic bs:TopicID="T1">
        <bs:text>XML et Office 2003</bs:text>
        <bs:topic bs:TopicID="T1.1">
            <bs:text>Visio 2003</bs:text>
        </bs:topic>
        <bs:topic bs:TopicID="T1.2">
            <bs:text>Project 2003</bs:text>
        </bs:topic>
        <bs:topic bs:TopicID="T1.3">
            <bs:text>OneNote 2003</bs:text>
        </bs:topic>
        <bs:topic bs:TopicID="T1.4">
            <bs:text>Access 2003</bs:text>
        </bs:topic>
        <bs:topic bs:TopicID="T1.5">
            <bs:text>Word 2003</bs:text>
        </bs:topic>
        <bs:topic bs:TopicID="T1.6">
            <bs:text>XSD/DTD/XSL</bs:text>
        </bs:topic>
        <bs:topic bs:TopicID="T1.7">
            <bs:text>FrontPage 2003</bs:text>
        </bs:topic>
        <bs:topic bs:TopicID="T1.8">
            <bs:text>XMLSpy</bs:text>
        </bs:topic>
    </bs:topic>
    <bs:topic bs:TopicID="T2">
        <bs:text>Voilà l'import</bs:text>
    </bs:topic>
</bs:Brainstorm>
```

et importons dans MS Visio:



Résultat:



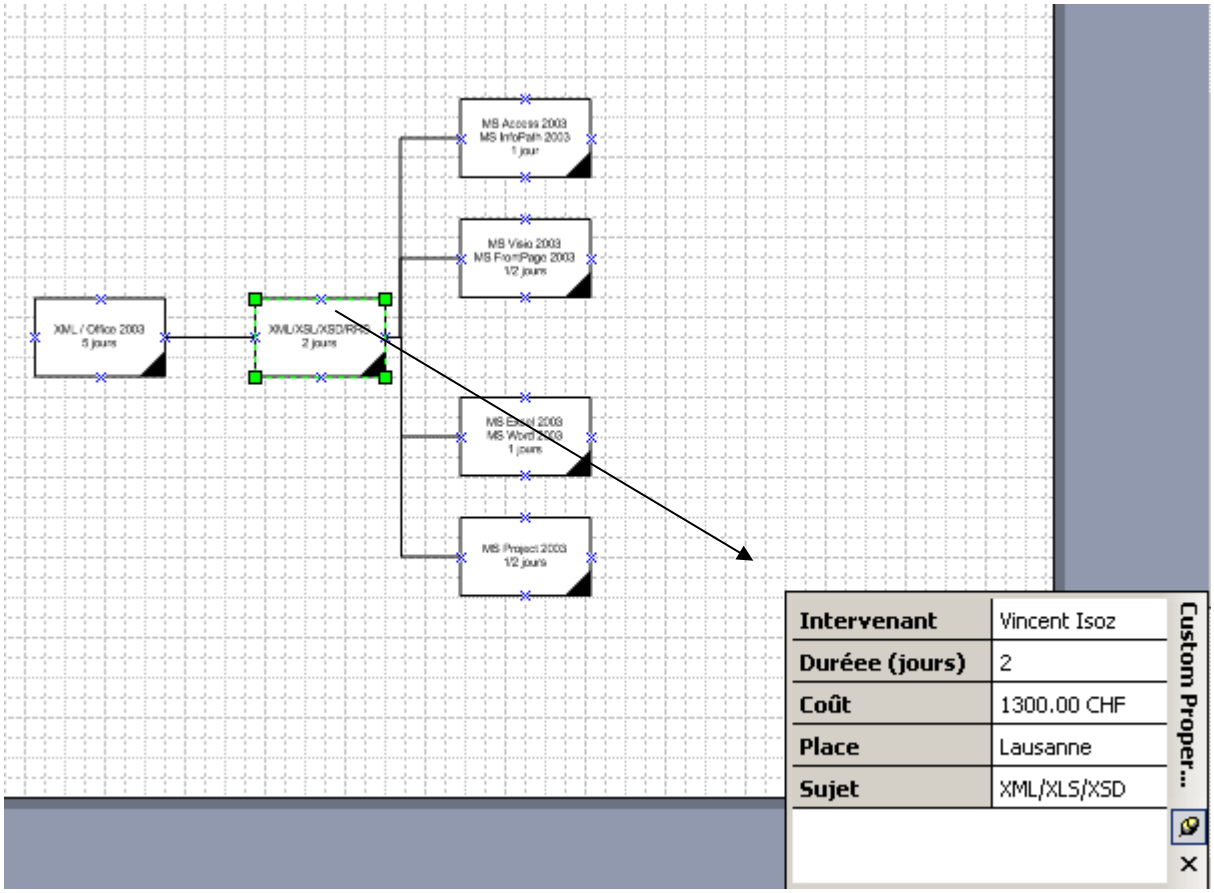
Comme quoi cela marche parfaitement dans les deux sens.

10.7.2 Rapports XML (exemple 2)

Voici mon préféré (je l'utilise lors de mes interventions d'audit dans des grandes entreprises internationales dont je ne citerai pas les noms).

Un reporting d'analyse chiffré d'un Businness Process en XML:

Soit à considérer le Processus (c'est du n'importe quoi mais c'est juste un exemple) suivant avec des propriétés identiques pour chaque forme comme indiqué dans la capture d'écran ci-dessous:

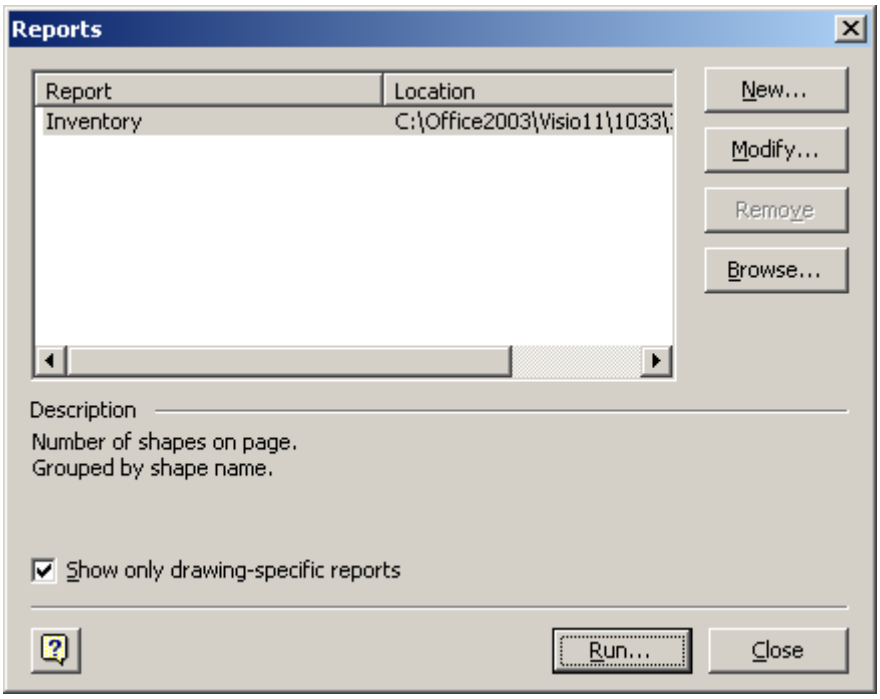


Le but: créer un rapport XML des coûts de l'un des chemins du graphe du processus.

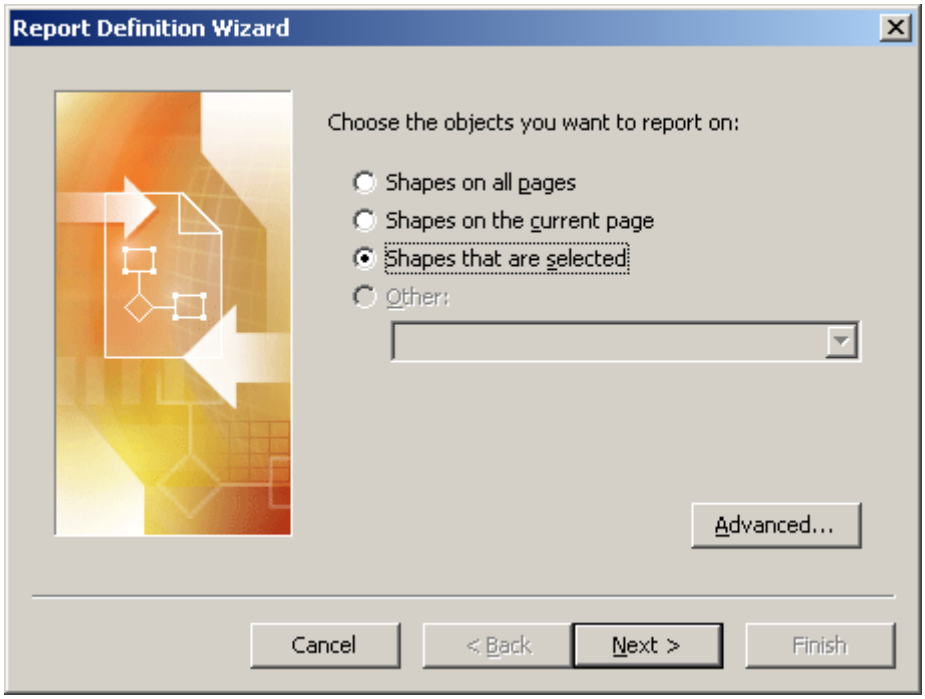
Méthode (sans entrer dans toutes les possiblités que cet outil propose):

1. Sélectionner le chemin (éléments du processus) intéressé par le reporting

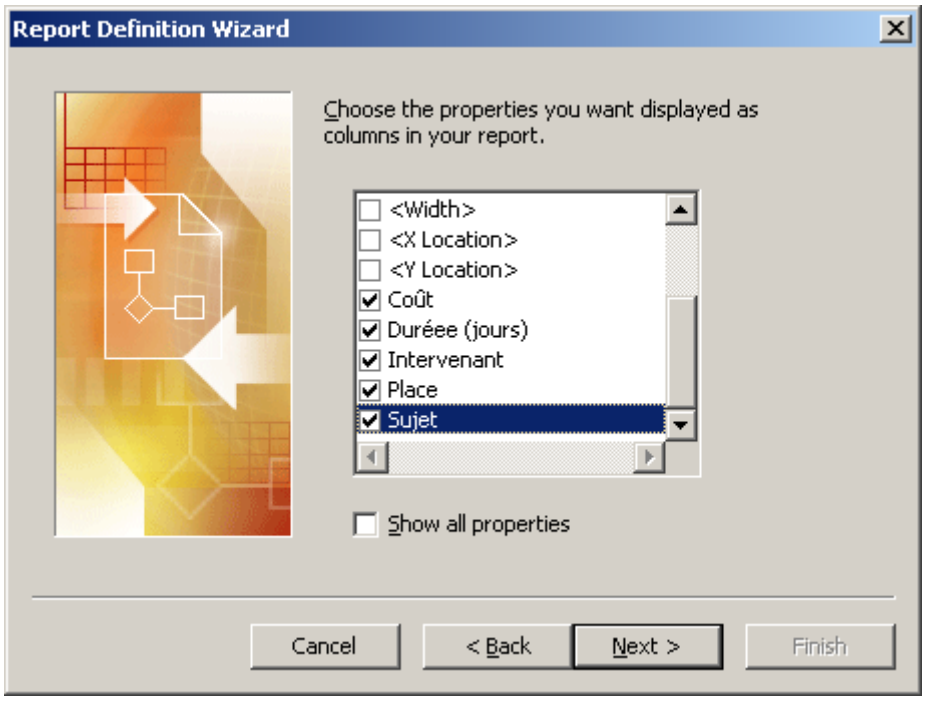
2. Aller dans le menu *Tools/Report*:



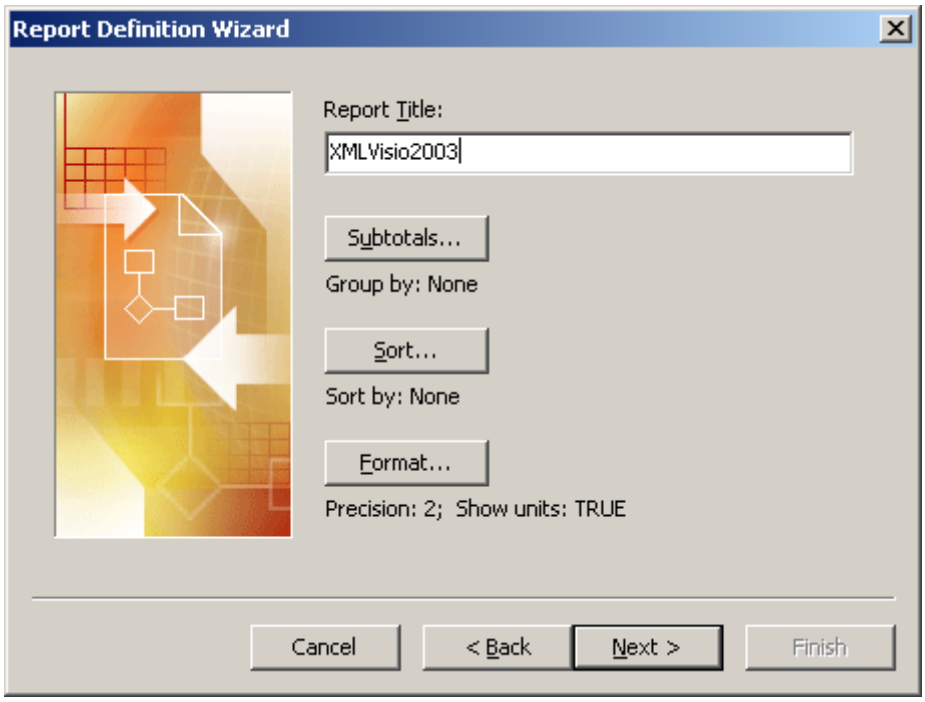
Cliquez sur *New* et sélectionnez *Shapes that are selected* et cliquez sur *Next*:



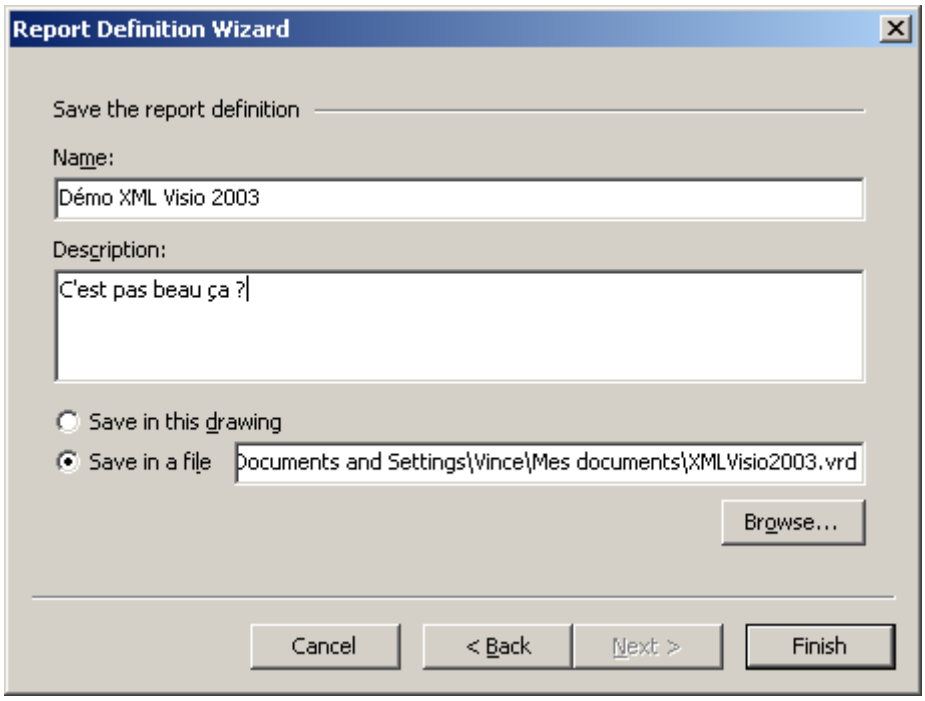
et cliquez sur *Next* et cochez ce que vous voulez dans votre rapport (afin de faire des "statistiques" par exemples sur les valeurs numériques):

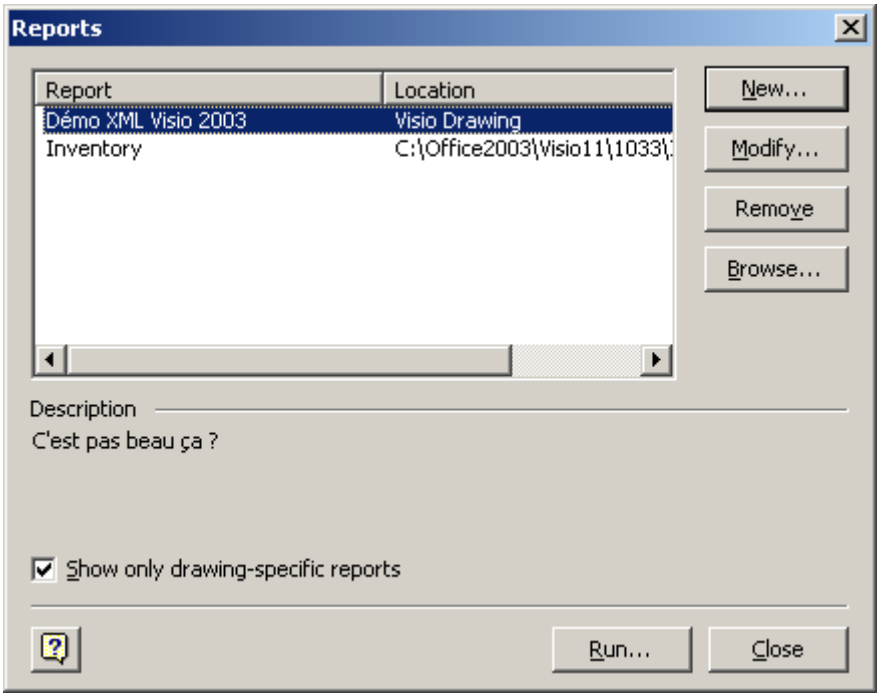


Cliquez sur *Next* et donnez un titre (=/ nom) à votre rapport (ne sélectionnez rien d'autre car nous ne souhaitons pas dans cette démo faire de Tris, Filtres ou Calculs à ce niveau du discours):

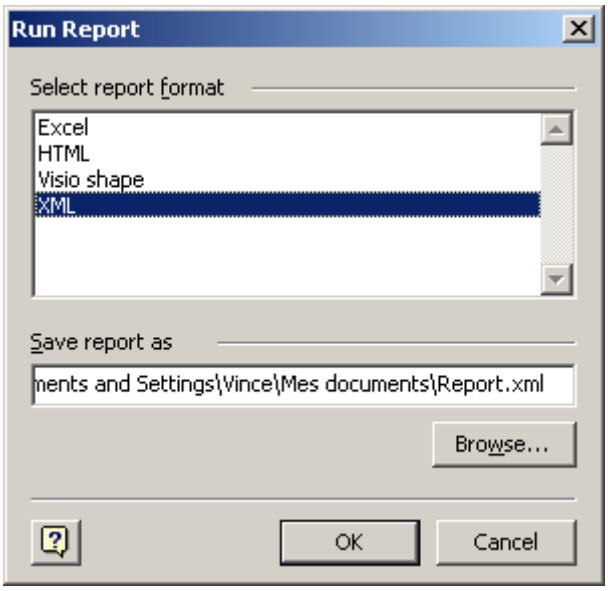


Cliquez sur *Next*, donnez un nom à votre rapport, un descriptif (et au besoin enregistrez-le dans un fichier *.vrd que vous pourrez envoyer à vos collègues afin qu'ils puissent eux aussi générer le même type de rapport) et cliquez sur *Finish*:





Sélectionnez le rapport à effectuer et cliquez sur *Run* la fenêtre suivante apparaît alors (on vous laisse deviner que est le choix le plus puissant qui soit) et cliquez sur *OK*:



et vous obtiendrez un document XML très moche.

La raison en est relativement simple: le fichier résultant contient le schéma XSD (au début) et ensuite les données XML de notre rapport. Si nous supprimons la partie XSD cela allège déjà terriblement le fichier XML.

Ensuite, le traitement des données XML qui nous intéressent via XSL devient un exercice relativement simple (voir même trivial!) mais si nous souhaitons que l'utilisateur lambda puisse facilement modifier/lire le fichier XML vaut quand même mieux exporter les données par voie de programmation VBA.

10.8 MSN Messenger

Vous vous demandez certainement que vient faire MSN Messenger dans le suite MS Office 2003. Eh bien c'est simple:

Nous considérons MSN Messenger comme faisant partie intégrante de MS Office 2003 lorsque SharePoint Portal Server 2003 et Live Communication Server sont installés sur le serveur de l'entreprise (suivre une formation à ce sujet pour voir la valeur ajoutée de ces outils combinés)

Savez-vous que vous pouvez profiter des dernières actualités dans MSN Messenger 6.0: cet article vous explique comment faire.

Sympa, moi qui suis un aficionados de Messenger, je me pose tout de suite la question: est-ce possible de faire la même chose pour moi ?

La réponse est oui et c'est d'une simplicité enfantine.

Première étape, créez un petit logo de 30 par 30 pixels avec un fond transparent: Ensuite, créez une page HTML (ou dynamique) qui contiendra ce qui sera affiché dans Messenger. Et oui, une bête page HTML (xsl, php, asp, etc.). Pour ma part, pour mes tests, j'ai juste créé une page avec un DataGrid qui est lié avec les 10 derniers articles paru sur le site.

Avant dernière étape, saisissez le code XML suivant:

```
<tab>
 <image> http://www.c2i.fr/c2iTV/c2imsn.png</image>
 <name>c2i.fr</name>
 <tooltip>Les derniers articles c2i.fr</tooltip>
 <contenturl>http://www.c2i.fr/c2imsn.aspx </contenturl>
<hidden>false</hidden>
</tab>
```

Dans l'ordre on définit l'emplacement de l'image, le nom de l'onglet, son tooltip et l'URL du contenu. Fastoche.

Dernière et ultime étape, fermez MSN 6.0 puis allez dans le dossier caché contenant le fichier ConfigCache.xml. Il est dans:

Documents and Settings\nom utilisateur\Application Data\Microsoft\MSN Messenger\numéro de version de MSN Messenger

et ajoutez votre tab au sein des autres tab. Relancez MSN 6.0 (attention pour des raisons de sécurité, cela ne fonctionne plus sur d'autres systèmes) et voili, voilo, les infos de vote site dans Messenger ; -)

Microsoft à laissé tombé cette possibilité maintenant et c'est son service MSN Alerts utilisant des flux RSS qui le remplace. N'import qui peut s'abonner à un flux RSS (ou en créer un comme nous verrons) en passant par l'adresse:

http://signup.alerts.msn.com/alerts/editSignup.do?

11. XML et Javascript

Nous partons du fichier XML suivant:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
    <cd>
        <title>Empire Burlesque</title>
        <artist>Bob Dylan</artist>
        <country>USA</country>
        <company>Columbia</company>
        <price>10.90</price>
        <year>1985</year>
    </cd>
    <cd>
        <title>Hide your heart</title>
        <artist>Bonnie Tyler</artist>
        <country>UK</country>
        <company>CBS Records</company>
        <price>9.90</price>
        <year>1988</year>
    </cd>
    <cd>
        <title>Greatest Hits</title>
        <artist>Dolly Parton</artist>
        <country>USA</country>
        <company>RCA</company>
        <price>9.90</price>
        <year>1982</year>
    </cd>
    <cd>
        <title>Still got the blues</title>
        <artist>Gary Moore</artist>
        <country>UK</country>
        <company>Virgin records</company>
        <price>10.20</price>
```

Et du fichier XSL suivant:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th style="text-align:left">Title</th>
        <th style="text-align:left">Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Ensuite, dans le même dossier, nous créons le fichier *.html suivant:

```
<html>
<head>
<script>
function loadXMLDoc(filename){
if (window.ActiveXObject)
  {xhttp = new ActiveXObject("Msxml2.XMLHTTP");}
else
  {xhttp = new XMLHttpRequest();}
xhttp.open("GET", filename, false);
try {xhttp.responseType = "msxml-document"} catch(err) {} // Helping IE11
xhttp.send("");
return xhttp.responseXML;}

function displayResult(){
xml = loadXMLDoc("simple.xml");
xsl = loadXMLDoc("XStyleSheet.xsl");
// code for IE
if (window.ActiveXObject || xhttp.responseType == "msxml-document"){
  ex = xml.transformNode(xsl);
  document.getElementById("example").innerHTML = ex;}
// code for Chrome, Firefox, Opera, etc.
else if (document.implementation && document.implementation.createDocument){
  xsltProcessor = new XSLTProcessor();
  xsltProcessor.importStylesheet(xsl);
  resultDocument = xsltProcessor.transformToFragment(xml, document);
  document.getElementById("example").appendChild(resultDocument);}}
</script>
</head>
<body onload="displayResult()">
<div id="example"/>
</body>
</html>
```

Après avoir posé les fichiers sur un serveur, nous obtenons sur l'ordinateur actif:

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith
Tupelo Honey	Van Morrison
Soulsville	Jorn Hoel
The very best of	Cat Stevens
Stop	Sam Brown
Bridge of Spies	T`Pau
Private Dancer	Tina Turner
Midt om natten	Kim Larsen
Pavarotti Gala Concert	Luciano Pavarotti
The dock of the bay	Otis Redding
Picture book	Simply Red
Red	The Communards
Unchain my heart	Joe Cocker

12. XML et PHP

Jusqu'à maintenant nous avons vu comment traiter des fichiers XML avec du XSL les deux étant localisés au même point. Cependant, si nous avons un fichier XSL à disposition pour parser un document XML qui se trouve sur un serveur distant il faut procéder un peu autrement

12.1 Parser un fichier XML distant

Voici un fichier XML nommé simple.xml (que nous supposerons situé n'importe où sur le web) que nous poserons à la racine du serveur apache (*Program Files/EasyPHP/www*) installé pour le cours (ou IIS si le plug-in IIS/PHP est installé):

```
<?xml version="1.0" encoding="UTF-8"?>
<people>
    <person>
        <name>Jennifer Lopez</name>
        <email>jen(at)shanx.com</email>
    </person>
    <person>
        <name>Shashank Tripathi</name>
        <email>shanx(at)shanx.com</email>
    </person>
    <person>
        <name>Thievery Corporation</name>
        <email>thievery(at)shanx.com</email>
    </person>
</people>
```

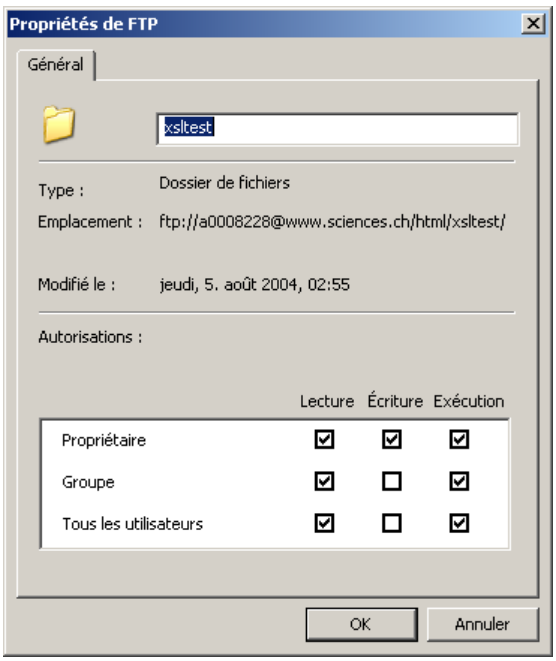
Voici son fichier XSL correspondant nommé simple.xsl (situé sur votre serveur):

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

   <html>
   <head><title>XSLT Testing</title>
   <style>
   .tiny {font-size: xx-small; font-family: verdana}
   </style>
   </head>

   <body>

   <h2>Parsing people...</h2>
   <div style="font-family:Arial,Sans-Serif">
   <xsl:apply-templates/>
   </div>
   </body>
   </html>

</xsl:template>

<xsl:template match="name">
<strong><span class="tiny">&#149;</span> <xsl:apply-templates/></strong><br />
</xsl:template>

<xsl:template match="email">
<xsl:apply-templates/><br /><br />
</xsl:template>

</xsl:stylesheet>
```

Rien de nouveau jusque là. Maintenant sur notre serveur nous créons le fichier PHP simple.php suivant (sans aller dans les détails):

```php
<?

// Xml and XSL files
$arguments = array(
     '/_xsl' => file_get_contents('simple.xsl'),
     '/_xml' => file_get_contents('simple.xml')
);

$xh = xslt_create();
$result = xslt_process($xh, 'arg:/_xml', 'arg:/_xsl', NULL, $arguments);

if (!$result) {
    // Something croaked. Show the error
    echo 'XSLT processing error: ' .xslt_error($xh) ;
}
else {
    // Output the resulting HTML
    echo $result;
}

// Destroy the XSLT processor
xslt_free($xh);

?>
```

Il suffit ensuite de mettre le dossier où se trouve le fichier PHP dans le mode suivant (sur le vrai serveur web, pas sur le apache en local sur la machin de dév. c'est inutile pour cette dernière):



ensuite, vous effectuez un clic droit sur l'icône d'EasyPHP près de l'horloge et vous allez dans

Configuration/Extensions PHP et vous activez l'extensions *php_xslt* et redémarrez ensuite le serveur Apache (au besoin activez toutes les extensions du XML) et le tour est joué (vous tapez votre adresse http://www.monsite.dns/simple.php et le tout est paré)

Le résultat étant:

Si le fichier XML se trouve ailleurs qu'en local il suffit de changer la ligne:

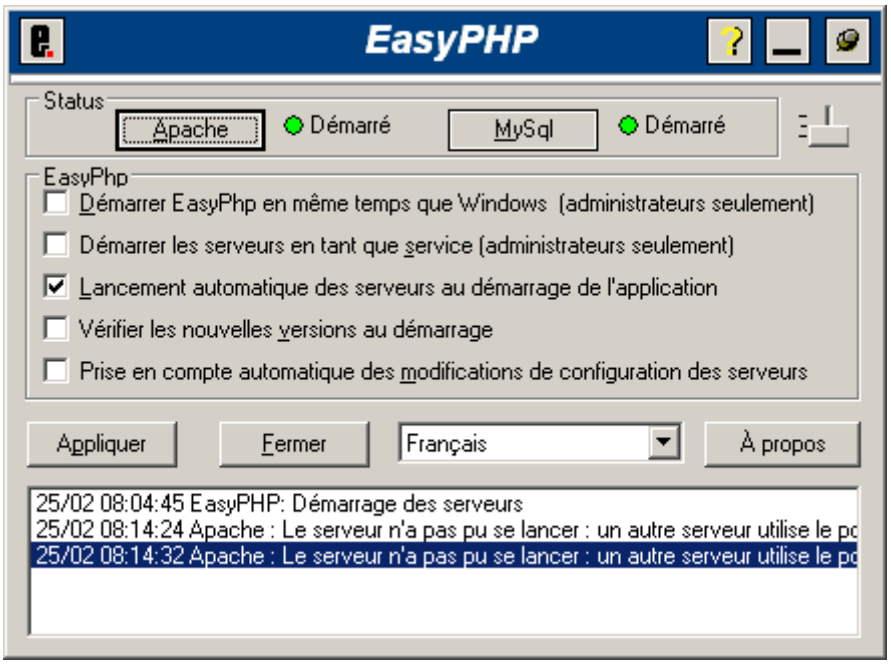
'/_xsl' => file_get_contents('simple.xsl'),

en mettant par exemple:

'/_xsl' => file_get_contents('http://www.autresite.dns/simple.xsl'),

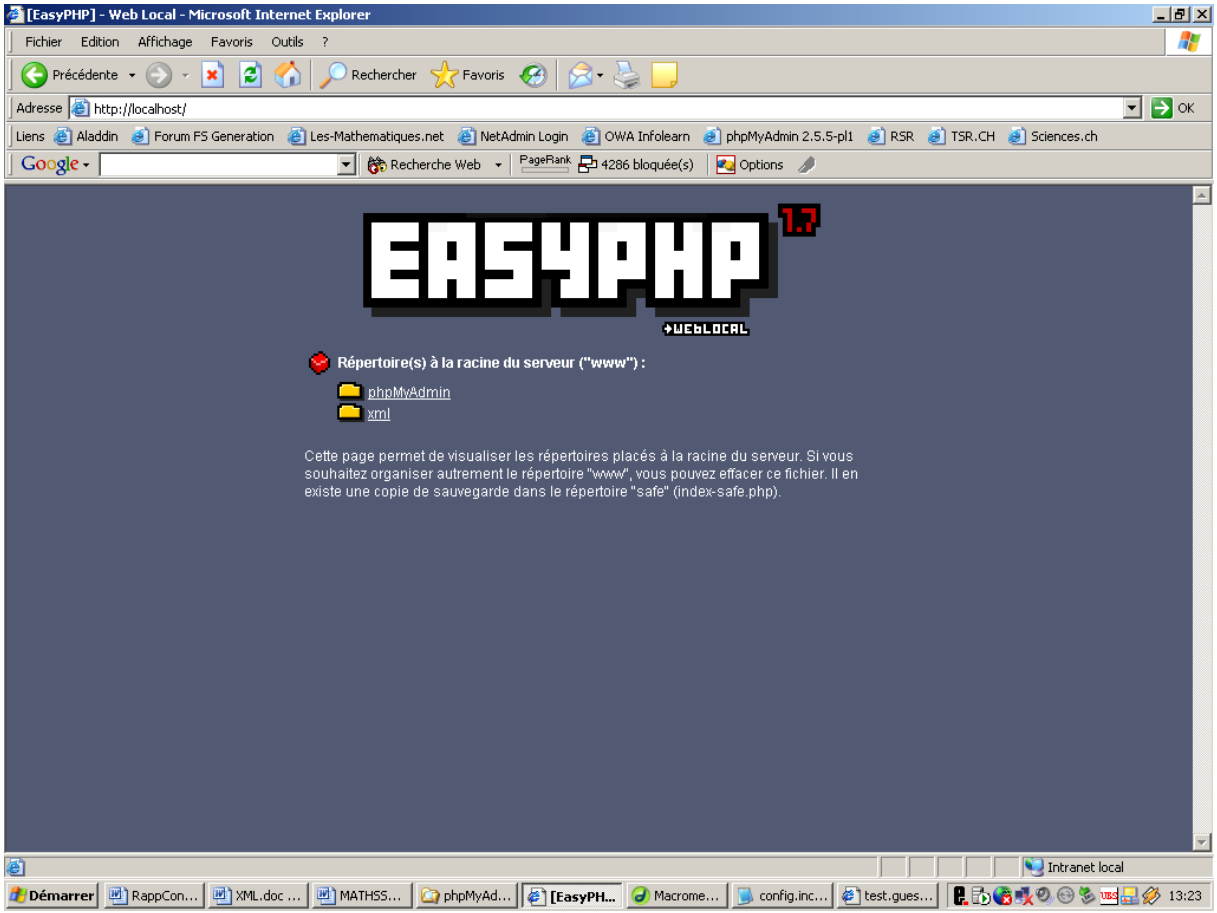
12.2 Construire un XML de mySQL

Pour cet exemple nous allons encore une fois nous servir de EasyPHP (1.7). Téléchargez et installez celui-ci sur votre machine. L'écran suivant devrait apparaître après un clic sur l'icône à coté de l'horloge dans la barre des tâches:

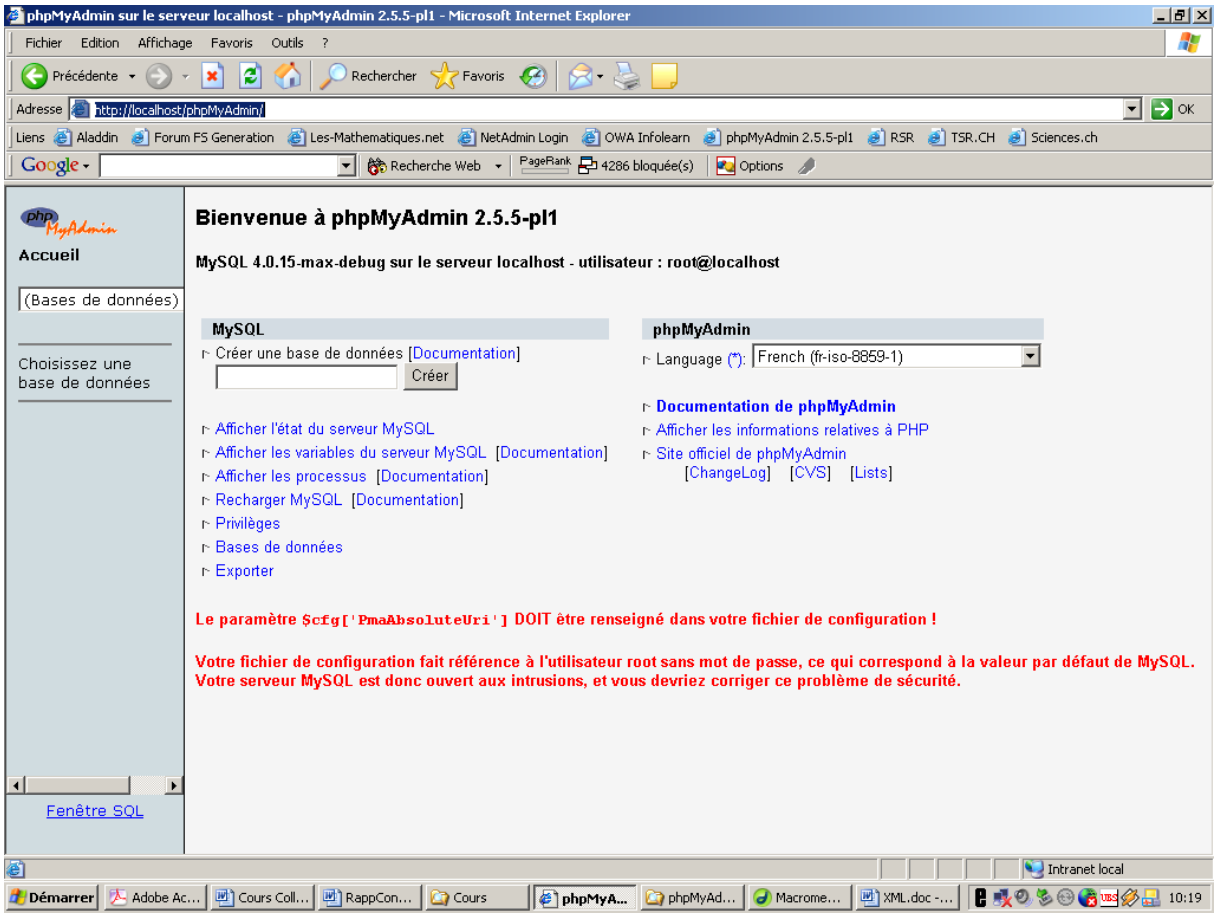


Remarque: si vous n'arrivez pas à démarrer votre EasyPHP, n'oubliez pas au préalable désactiver votre Firewall logiciel.

Si vous ouvrez internet Explorer et saisissez http://localhost vous devriez avoir (version 1.7 d'EasyPHP):



Téléchargez phpMyAdmin (www.phpmyadmin.net), mettez-le dans le dossier d'installation *www* de EasyPHP et ouvrez internet explorer à l'adresse: *http://localhost/phpMyAdmin/*

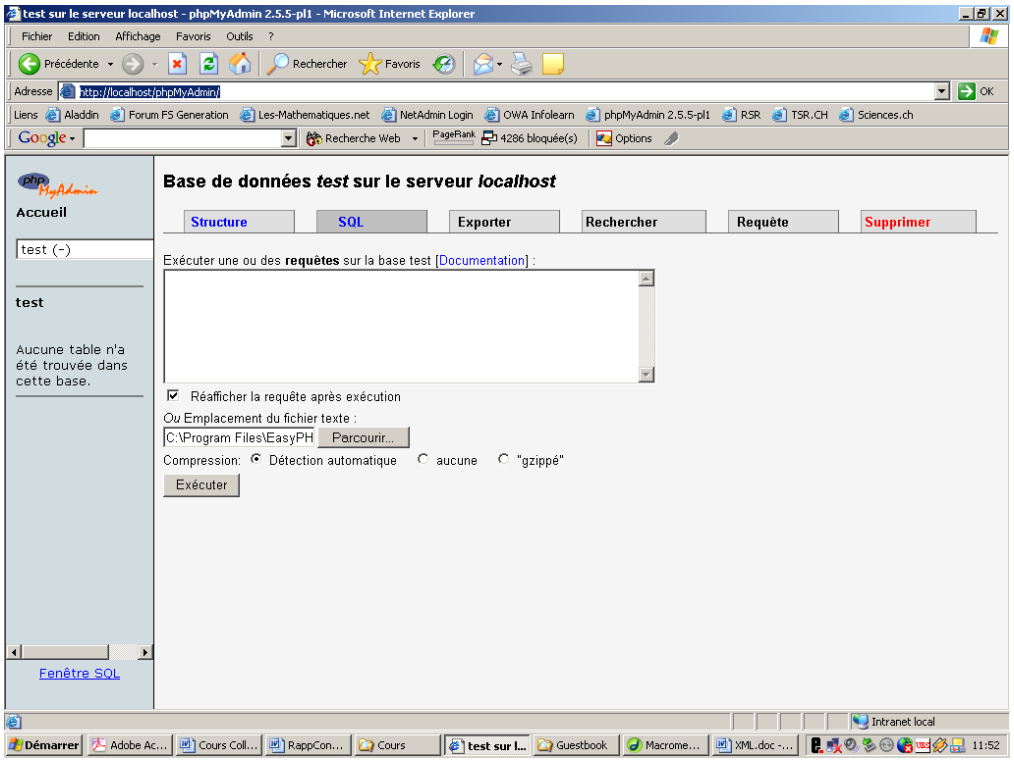


Lisez les messages en rouge et suivez la procédure indiquée dans le fichier congig.inc.php (normalement il suffit d'y avoir: $cfg['PmaAbsoluteUri'] = 'http://localhost/phpMyAdmin/';)

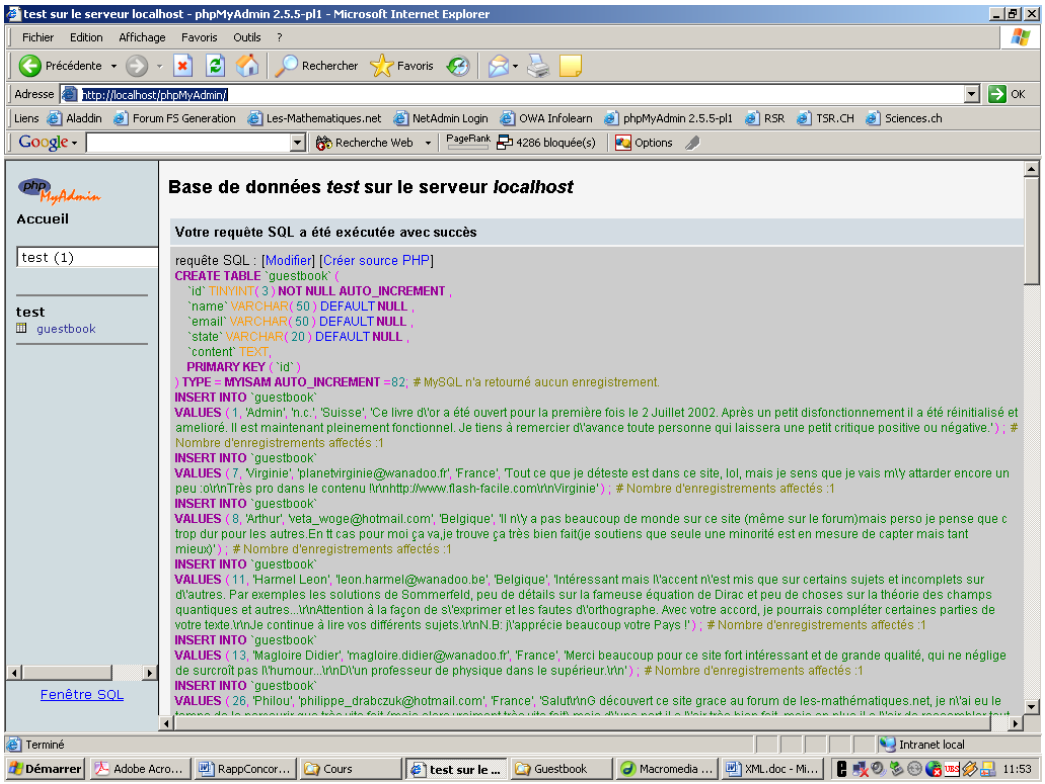
Ce message peut varier selon les verions de PHPMyAdmin… (les joies de l'informatique quoi…)

Maintenant dans PHPMyAdmin, créez une base vierge s'il n'en existe pas (maintenant avec la version 2.8.0.2 il en existe une par défaut nommée mySQL…)

Ensuite, parmi les fichiers d'exercices mis à disposition pour le cours, vous avez un fichier nommé *guestbook.sql.zip.sql*. Exécutez ce fichier dans la rubrique *SQL* de PHPmyAdmin (il s'agit simplement d'un petit livre d'or avec quelques données…):



et cliquez sur exécuter tel que vous obteniez:



Vous pouvez voir dans le volet de droite de l'explorateur, que ceci a créé une table nommée *guestbook* dans la base *test*.

Maintenant, l'exemple de script PHP suivant (avec les valeurs par défaut de PHPmyAdmin) se connecte à la base de données et exécute une requête (grand classique).

```
<?
$db_name = "test";
//attention le nom de la base peut changer suivant votre choix initial!
$connection = mysql_connect("localhost", "root", " ") or die("Connexion impossible.");
$table_name = 'guestbook';
```

Une fois la connexion établie, vous devez sélectionner la base de données courante à l'aide de la connexion MySQL. Le code suivant se charge de cette tâche:

```
$db = mysql_select_db($db_name, $connection);
```

À présent, rédigez une instruction SQL pour sélectionner toutes les lignes dans $table_name.

```
$query = "select * from " . $table_name;
```

Le cas échéant, vous pourrez ajouter des attributs ultérieurement. Pour l'instant, exécutez la requête de la manière suivante:

```
$result = mysql_query($query, $connection) or die("Impossible d'interroger la base de données");

$num = mysql_num_rows($result);
```

À ce stade, vous êtes prêt à créer un nouveau document XML. Il existe de nombreuses manières de s'y prendre, mais je pense que l'approche utilisée dans le listing plus base s'adaptera à la plupart des situations.

Voici ce qui se produit, étape par étape. La variable num montre la présence de données de ligne (row) dans votre requête, mesurables à l'aide de la fonction mysql_num_rows de MySQL. Cela nous mène à votre sortie procédurale du XML. La variable $file contient un pointeur vers l'objet fichier généré lorsque PHP réussit à lire le système de fichiers à la recherche de results.xml. S'il trouve results.xml, votre objet fichier PHP, nommé file, est créé et rendu accessible en écriture. Maintenant, vous pouvez y imprimer les contenus d'une variable, ce que vous ferez car les permissions définies pour votre répertoire autorisent PHP à le faire.

N'oubliez pas que, pour des raisons de sécurité, cet exemple n'est pas à suivre dans les applications web réelles. Pour vous assurer que l'implémentation des concepts abordés dans cet article est sûre, vous devez fournir un chemin complet vers un répertoire contenant les fichiers que vous souhaitez ouvrir en écriture et vérifier qu'il figure dans un répertoire situé au-dessus de votre racine web.

Ensuite, la fonction mysql_fetch_array de PHP convertit la variable de requête $result en un tableau et effectue une boucle sur ses clés. Si pageTitle figurait parmi les colonnes renvoyées dans la requête, pour chaque ligne renvoyée, du texte est écrit au format XML sur la variable de chaîne $_xml.

Notez que l'opérateur ".=", qui sert à ajouter les chaînes au format XML sous forme de valeurs, est lu à partir de $row. Lorsque la boucle est terminée, le nœud XML racine est imprimé sur la variable $_xml; la variable dans son intégralité est ensuite écrite dans file.xml à l'aide de la fonction fwrite de PHP.

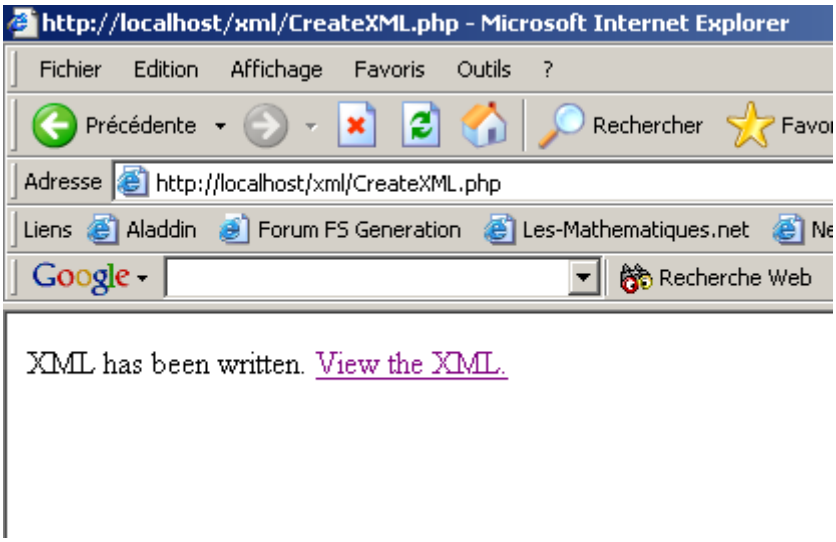
À ce stade, un lien apparaît à l'écran. Assurez-vous que ce lien pointe sur le chemin de votre fichier XML, sinon vous ne pourrez pas voir le XML mis en forme produit par PHP à partir de votre requête MySQL.

```
if ($num != 0) {
 $file= fopen("results.xml", "w");
 $_xml ="<?xml version=\"1.0\" encoding=\"ISO-8859-1\" ?>\r\n";
 $_xml .="<site>\r\n";
 while ($row = mysql_fetch_array($result)) {
 if ($row["name"]) {
 $_xml .="\t<page title=\"" . $row["name"] . "\">\r\n";
 $_xml .="\t\t<file>" . $row["content"] . "</file>\r\n";
$_xml .="\t</page>\r\n";
 } else {
 $_xml .="\t<page title=\"Nothing Returned\">\r\n";
$_xml .="\t\t<file>none</file>\r\n";
 $_xml .="\t</page>\r\n";
 } }
 $_xml .="</site>";
 fwrite($file, $_xml);
 fclose($file);
 echo "XML has been written.  <a href=\"results.xml\">View the XML.</a>";
 } else {
 echo "No Records found";
 } ?>
```

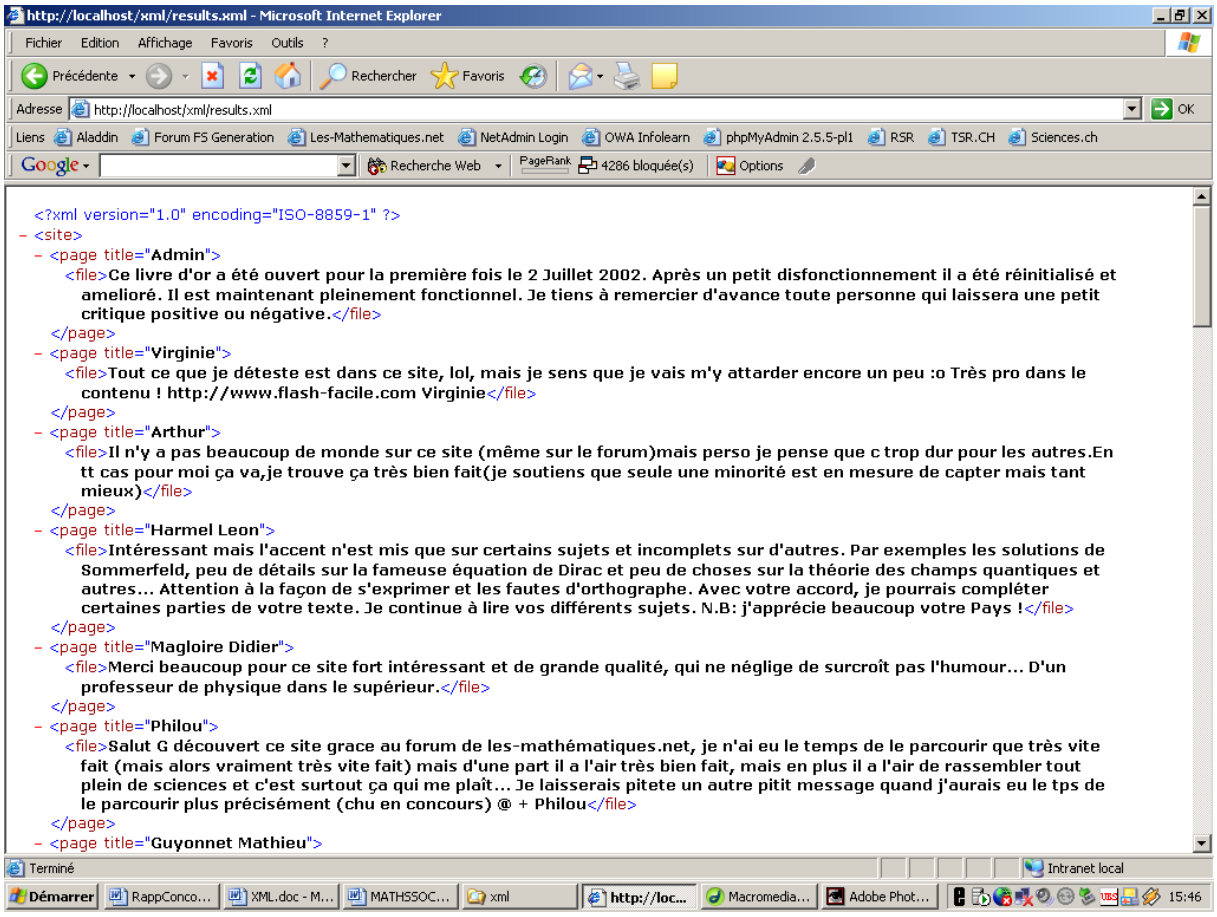
L'ensemble du code devrait ressembler à:

```php
<?
$db_name = "test";
$connection = mysql_connect("localhost", "root", "") or die("Connexion impossible.");
$table_name = 'guestbook';

$db = mysql_select_db($db_name, $connection);
$query = "select * from " . $table_name;
$result = mysql_query($query, $connection) or die("Impossible d'interroger la base de données");
$num = mysql_num_rows($result);

if ($num != 0) {
 $file= fopen("results.xml", "w");
 $_xml ="<?xml version=\"1.0\" encoding=\"ISO-8859-1\" ?>\r\n";
 $_xml .="<site>\r\n";
 while ($row = mysql_fetch_array($result)) {
 if ($row["name"]) {
 $_xml .="\t<page title=\"" . $row["name"] . "\">\r\n";
 $_xml .="\t\t<file>" . $row["content"] . "</file>\r\n";
$_xml .="\t</page>\r\n";
 } else {
 $_xml .="\t<page title=\"Nothing Returned\">\r\n";
$_xml .="\t\t<file>none</file>\r\n";
 $_xml .="\t</page>\r\n";} }
 $_xml .="</site>";
 fwrite($file, $_xml);
 fclose($file);
 echo "XML has been written.  <a href=\"results.xml\">View the XML.</a>"; } else {
 echo "No Records found";
 } ?>
```

Enregistrez ensuite ce code sous le nom *CreateXML.php* dans le répertoire *www* de EasyPHP et saisissez dans le navigateur IE l'adresse suivante http://localhost/ *CreateXML.php*:



Si tout se passe bien, vous verrez dans votre répertoire un nouveau fichier appelé results.xml. Il s'agit du fichier XML créé avec le code PHP. Si vous cliquez sur le lien ci-dessus, vous devriez avoir:


http://localhost/xml/results.xml - Microsoft Internet Explorer
Fichier Edition Affichage Favoris Outils ?
Précédente Rechercher Favoris
Adresse http://localhost/xml/results.xml
OK
Liens Aladdin Forum FS Generation Les-Mathematiques.net NetAdmin Login OWA Infolearn phpMyAdmin 2.5.5-pl1 RSR TSR.CH Sciences.ch
Google Recherche Web PageRank 4286 bloquée(s) Options
<?xml version="1.0" encoding="ISO-8859-1" ?>
<site>
<page title="Admin">
<file>Ce livre d'or a été ouvert pour la première fois le 2 Juillet 2002. Après un petit disfonctionnement il a été réinitialisé et amelioré. Il est maintenant pleinement fonctionnel. Je tiens à remercier d'avance toute personne qui laissera une petit critique positive ou négative.</file>
</page>
<page title="Virginie">
<file>Tout ce que je déteste est dans ce site, lol, mais je sens que je vais m'y attarder encore un peu :o Très pro dans le contenu ! http://www.flash-facile.com Virginie</file>
</page>
<page title="Arthur">
<file>Il n'y a pas beaucoup de monde sur ce site (même sur le forum)mais perso je pense que c trop dur pour les autres.En tt cas pour moi ça va,je trouve ça très bien fait(je soutiens que seule une minorité est en mesure de capter mais tant mieux)</file>
</page>
<page title="Harmel Leon">
<file>Intéressant mais l'accent n'est mis que sur certains sujets et incomplets sur d'autres. Par exemples les solutions de Sommerfeld, peu de détails sur la fameuse équation de Dirac et peu de choses sur la théorie des champs quantiques et autres... Attention à la façon de s'exprimer et les fautes d'orthographe. Avec votre accord, je pourrais compléter certaines parties de votre texte. Je continue à lire vos différents sujets. N.B: j'apprécie beaucoup votre Pays !</file>
</page>
<page title="Magloire Didier">
<file>Merci beaucoup pour ce site fort intéressant et de grande qualité, qui ne néglige de surcroît pas l'humour... D'un professeur de physique dans le supérieur.</file>
</page>
<page title="Philou">
<file>Salut G découvert ce site grace au forum de les-mathématiques.net, je n'ai eu le temps de le parcourir que très vite fait (mais alors vraiment très vite fait) mais d'une part il a l'air très bien fait, mais en plus il a l'air de rassembler tout plein de sciences et c'est surtout ça qui me plaît... Je laisserais pitete un autre pitit message quand j'aurais eu le tps de le parcourir plus précisément (chu en concours) @ + Philou</file>
</page>
<page title="Guyonnet Mathieu">
Terminé
Intranet local
Démarrer RappConco... XML.doc - M... MATHSSOC... xml http://loc... Macromedia... Adobe Phot... 15:46

13. XML et ASP 3.0

A nouveau nous allons faire de même qu'en PHP mais ce coup-ci avec de l'ASP 3.0 et en faisant usage de IIS (vous devez mettre les fichiers dans le dossier wwwroot du dossier InetPub pour que cela marche).

Voici notre document XML de départ:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<news>
    <titre>Valider une adresse email</titre>
    <categorie>ASP TRUCS et ASTUCES</categorie>
    <resumer>Le code suivant vous permet de tester la validité d'une adresse email</resumer>
    <contenu>Pour valider une adresse email ...</contenu>
    <auteur>Jean-christophe</auteur>
</news>
```

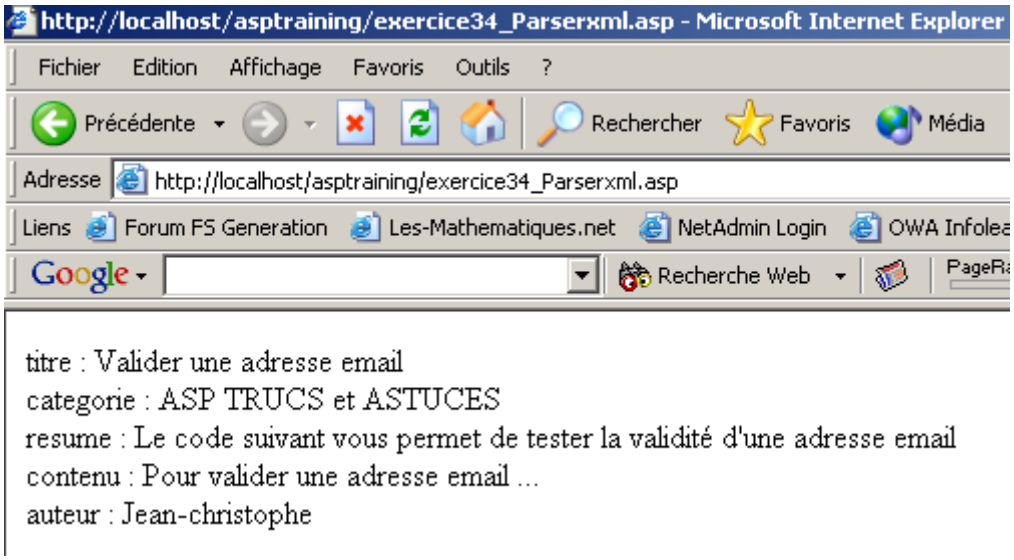
Sans entrer dans les détails voici le document ASP 3.0 qui va parser notre XML:

```
<% @Language=VBScript %>
<% Option Explicit %>
<HTML>
<BODY>

<%
    Dim xmlDoc
    Dim titre,categorie,contenu,auteur,resumer
    Set xmlDoc = Server.CreateObject("Microsoft.XMLDOM")

    if (xmlDoc.load (Server.MapPath(".")&"\DocumentParser.xml")) then
        titre = xmlDoc.documentElement.childNodes(0).text
        categorie = xmlDoc.documentElement.childNodes(1).text
        resumer = xmlDoc.documentElement.childNodes(2).text
        contenu = xmlDoc.documentElement.childNodes(3).text
        auteur = xmlDoc.documentElement.childNodes(4).text
    else
        set xmlDoc = Nothing
        response.write("Une erreur s'est produite pendant le chargement du fichier XML")
        response.end
    end if
    set xmlDoc =Nothing

    '--- On génère la sortie HTML

    response.write ("titre :  " & titre & "<br>")
    response.write ("categorie : " & categorie & "<br>")
    response.write ("resume :  " & resumer & "<br>")
    response.write ("contenu : " & contenu & "<br>")
    response.write ("auteur :  " & auteur & "<br>")

%>
</BODY>
</HTML>
```

Remarque: Pour accéder à un attribut nous écrirons:

xmlDoc.documentElement.childNodes(0).attributes.getNamedItem("code").nodeValue

Ce qui donne:



Nous pouvons aussi faire un peu mieux que cela… soit notre flux RSS que nous avions utilisé comme exemple plus haut dans ce support:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml:stylesheet type="text/xsl" href="RSStyleSheet.xsl"?>
<rss version="2.0">
    <channel>
        <title>Sciences.ch</title>
        <link>http://www.sciences.ch/xmlrss/index.php</link>
        <category>Mathématique, Physique théorique</category>
        <description>
Se voulant un complément aux études scolaires, Sciences.ch ce propose d'aborder différents domaines des mathématiques et (
        </description>
        <language>fr</language>
        <copyright>Concept, Design, Contenu, 2002-2004 Sciences.ch. Tous droits réservés</copyright>
        <image>
            <title>Sciences.ch</title>
            <url>http://www.sciences.ch/images/bansciencech.gif</url>
            <link>http://www.sciences.ch</link>
            <width>88</width>
            <height>31</height>
            <description>Au coeur de la Science!</description>
        </image>
        <item>
            <mois>052004</mois>
            <title>Origine de la chaleur</title>
            <Auteur>V. Isoz</Auteur>
            <description>Qu'est-ce que la chaleur ? Voici une question que se posent nombre d'étudiants et parfois suffisamme
            <link>http://www.sciences.ch/htmlfr/mecanique/mecanthermodyn01.php#chaleur</link>
            <pubDate>Je, 27 Mai 2004 11:50:01 GMT</pubDate>
        </item>
        <item>
            <mois>052004</mois>
            <title>Théorème de Toricelli</title>
            <Auteur>V. Isoz</Auteur>
            <description>Dans le cadre de la mécanique des milieux continus (M.M.C), nous pouvons nous poser un très grand
            <link>http://www.sciences.ch/htmlfr/mecanique/mecanfluides01.php#toricelli</link>
        </item>
        <item>
            <mois>052004</mois>
            <title>Effet Venturi</title>
            <Auteur>V. Isoz</Auteur>
            <description>Dans le cadre des écoulements des liquides dans des volumes simples (des tubes typiquement...), no
            <link>http://www.sciences.ch/htmlfr/mecanique/mecanfluides01.php#effetventuri</link>
```

ce fichier XML sera supposé enregistré sur un serveur web distant ou local (http://localhost/rss.xml dans notre exemple !).

Voici le code ASP pour le lire:

```
<%@ Language="VBScript" %>
<% Option Explicit %>
<% Response.Charset = "iso-8859-1" %>
<html>
<head>
<title>ASP RSS Feed Reader</title>
</head>
<body>
<%

    Dim objXML
    Dim objItemList
    Dim objItem
    Dim strHTML

    Set objXML = Server.CreateObject("MSXML2.FreeThreadedDOMDocument")
    objXML.async = False
    objXML.setProperty "ServerHTTPRequest", True
    objXML.Load("http://www.excelsia.ch/sps/rss/enews.xml")
    Set objItemList = objXML.getElementsByTagName("item")
    Set objXML = Nothing

    For Each objItem In objItemList
       ' Feed childNodes: 0=title, 1=link, 2=description
       strHTML = strHTML & "<p>" & vbCrLf
       strHTML = strHTML & "<a href=""" & objItem.childNodes(1).text & """>"
       strHTML = strHTML & "<strong><em>" & objItem.childNodes(0).text
       strHTML = strHTML & "</em></strong></a><br />" & vbCrLf
       strHTML = strHTML & Replace(objItem.childNodes(2).text, "<br>", "<br />") & vbCrLf
       strHTML = strHTML & "</p>" & vbCrLf
    Next

    Set objItemList = Nothing
    response.write strHTML
%>

</body>
</html>
```

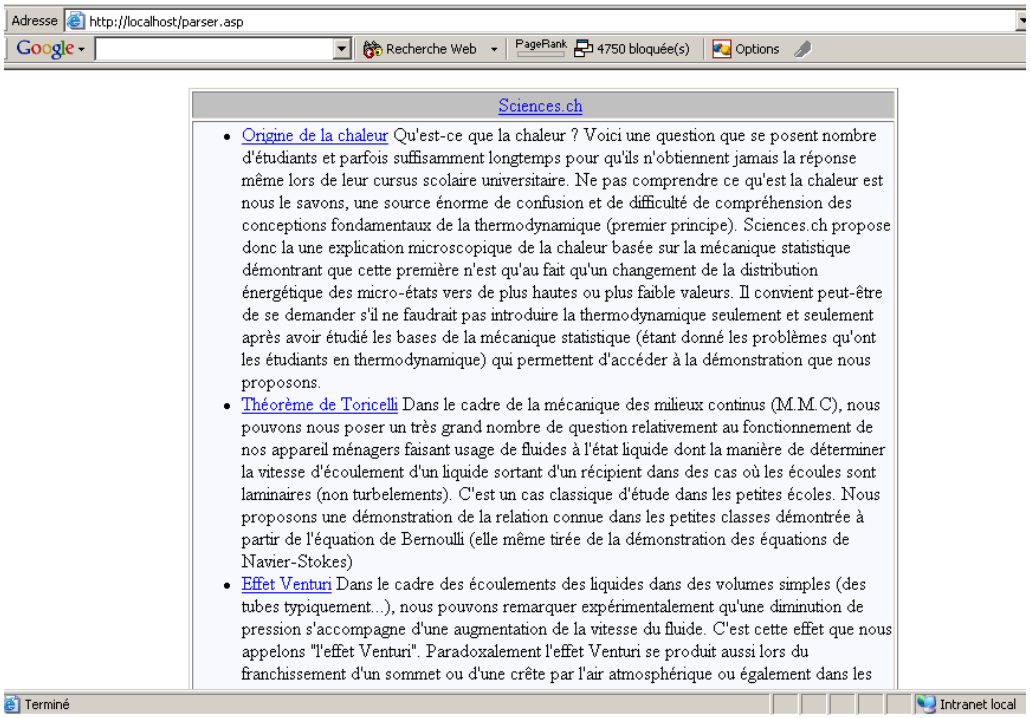
Encore mieux… soit le fichier XSL suivant, nommé *news.xsl* et enregistré sur le Localhost de IIS, configuré pour parser n'importe quel fichier utilisant la norme RSS 2.0:

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" encoding="iso-8859-1" omit-xml-declaration="yes" indent="yes"/>
<xsl:template match="*">
<table border="1" width="600" align="center">
<tr><td valign="top" align="center" class="title" bgcolor="silver" >
<a>
<xsl:attribute name="href">
<xsl:value-of select="*[local-name()='channel']/*[local-name()='link']"/>
</xsl:attribute>
<xsl:attribute name="target">
<xsl:text>top</xsl:text>
</xsl:attribute>
<xsl:value-of select="*[local-name()='channel']/*[local-name()='title']" disable-output-escaping="yes"/>
</a>
<xsl:text disable-output-escaping="yes">&amp;nbsp;</xsl:text>
<xsl:value-of select="*[local-name()='channel']/*[local-name()='lastBuildDate']"/>
</td></tr><tr><td valign="top" bgcolor="ghostwhite" class="headlines" >
<ul>
<xsl:for-each select="//*[local-name()='item']">
<li>
<a>
<xsl:attribute name="href">
<xsl:value-of select="*[local-name()='link']"/>
</xsl:attribute>
<xsl:attribute name="target">
<xsl:text>top</xsl:text>
</xsl:attribute>
<xsl:value-of select="*[local-name()='title']" disable-output-escaping="yes"/>
</a>
<xsl:text disable-output-escaping="yes">&amp;nbsp;</xsl:text>
<xsl:value-of select="*[local-name()='description']" disable-output-escaping="yes"/>
</li>
</xsl:for-each>
</ul>
</td></tr>
</table>
</xsl:template>
<xsl:template match="/">
<xsl:apply-templates/>
</xsl:template>
</xsl:stylesheet>
```

Voici le fichier ASP 3.0 correspondant qui va allé chercher le flux et parser ce premier avec le fichier XSL. Le tout est appelé par une sub que nous appelerons *getXML*:

```
<% @Language="VBScript" %>
<%
Sub getXML(sourceFile)
     dim styleFile
     dim source, style
     styleFile = Server.MapPath("news.xsl")

     set source = Server.CreateObject("Msxml2.DomDocument")
     source.async = false
     source.setProperty "ServerHTTPRequest", true
     source.load CStr(sourceFile)

     set style = Server.CreateObject("Msxml2.DomDocument")
     style.async = false
     style.load styleFile

     source.transformNodeToObject style, Response
     set source = nothing
     set style = nothing
End Sub
%>
<html>
<body>
<% getXML("http://localhost/rss.xml") %>
</body>
</html>
```

Le tout dans Ineternet Explorer donnera:



14. XML et VB.NET

Nous allons voir maintenant comment parser traiter également un flux RSS en VB.Net à partir d'un fichier XSL.

Nous partons d'abord du fichier XSL très sommaire suivant:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" />

  <xsl:template match="/">
    <xsl:for-each select="rss/channel/item">
      <p>
      <a href="{link}"><strong><xsl:value-of select="title" /></strong></a>
      <xsl:value-of disable-output-escaping="yes" select="description" />
      </p>
      <hr />
    </xsl:for-each>
  </xsl:template>

</xsl:stylesheet>
```

Qui sera appelé par la page ASPX suivante pour afficher le résultat:

Remarque: Le *disable-output-espacing* est présent pour jouer avec des caractères spéciaux se trouvant dans des CDATA.

Voici le fichier VB.Net:

```
<%@ Page Language="VB" Debug="False" %>
<%@ Import Namespace="System.IO" %>
<%@ Import Namespace="System.Xml" %>
<%@ Import Namespace="System.Xml.Xsl" %>
<script language="VB" runat=server>
  Sub Page_Load(sender As Object, e As EventArgs)

    Dim strXmlSrc  As String = "http://www.excelsia.ch/htmlfr/videos/dnews.xml"
    Dim strXslFile As String = Server.MapPath("VBXSL.xsl")
    Dim myXmlDoc As XmlDocument = New XmlDocument()
    myXmlDoc.Load(strXmlSrc)
    Dim myXslDoc As XslTransform = New XslTransform()
    myXslDoc.Load(strXslFile)
    ' Create a StringBuilder and then point a StringWriter at it.
    ' We'll use this to hold the HTML output by the Transform method.
    Dim myStringBuilder As StringBuilder = New StringBuilder()
    Dim myStringWriter  As StringWriter  = New StringWriter(myStringBuilder)
    ' Call the Transform method of the XslTransform object passing it
    ' our input via the XmlDocument and getting output via the StringWriter.
    myXslDoc.Transform(myXmlDoc, Nothing, myStringWriter)
    ' Take our resulting HTML and display it via an ASP.NET
    ' literal control.
    RssHtml.Text = myStringBuilder.ToString

  End Sub
</script>
<html>
<head>
  <title>VB ASP.NET RSS Feed Reader</title>
</head>
<body>
<asp:Literal id="RssHtml" runat="server" />
</body>
</html>
```

15. XML et C#

De même en C# mais cette fois-ci sans XSL:

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
    public void ProcessRSSItem(string rssURL)
    {
        System.Net.WebRequest myRequest = System.Net.WebRequest.Create(rssURL);
        System.Net.WebResponse myResponse = myRequest.GetResponse();

        System.IO.Stream rssStream = myResponse.GetResponseStream();
        System.Xml.XmlDocument rssDoc = new System.Xml.XmlDocument();
        rssDoc.Load(rssStream);

        System.Xml.XmlNodeList rssItems = rssDoc.SelectNodes("rss/channel/item");

        string title = "";
        string link = "";
        string description = "";

        for (int i = 0; i < rssItems.Count; i++)
        {
            System.Xml.XmlNode rssDetail;

            rssDetail = rssItems.Item(i).SelectSingleNode("title");
            if (rssDetail != null)
            {
                title = rssDetail.InnerText;
            }
            else
            {
                title = "";
            }

            rssDetail = rssItems.Item(i).SelectSingleNode("link");
            if (rssDetail != null)
            {
                link = rssDetail.InnerText;
            }
            else


            {
                link = "";
            }

            rssDetail = rssItems.Item(i).SelectSingleNode("description");
            if (rssDetail != null)
            {
                description = rssDetail.InnerText;
            }
            else
            {
                description = "";
            }

            Response.Write("<p><b><a href='" + link + "' target='new'>" + title + "</a></b><br/>");
            Response.Write(description + "</p>");
        }
    }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>C# ASP.Net RSS Reader</title>
</head>
<body>
            <%
                string rssURL = "http://www.codeguru.com/icom_includes/feeds/codeguru/rss-all.xml";
                Response.Write("<font size=5><b>Site: " + rssURL + "</b></font><Br />");
                ProcessRSSItem(rssURL);
                Response.Write("<hr />");
            %>
</body>
</html>
```

16. XML et ADOBE ACROBAT 6.0

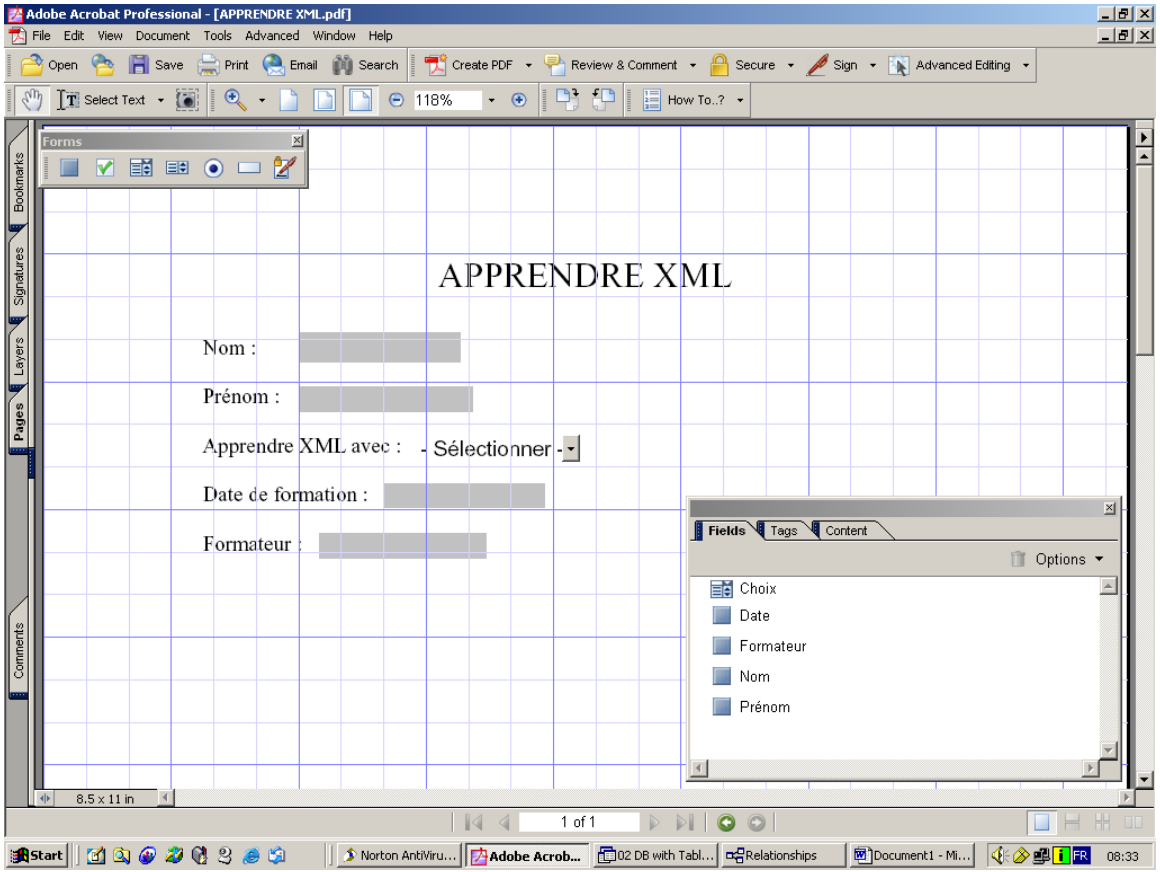
Le mixe entre Adobe Acrobat 6.0 XML et MS Office 2003 est la fonctionnalité qui me plaît le plus. La raison est due à deux faits:

1. Quasiment tous les ordinateurs sont équipés d'Adobe Reader
2. Les formulaires au format PDF sont dynamiques, esthétiques et légers
3. Le code XML généré par Adobe Acrobat est d'une extrême sobriété par rapport à MS Visio 2003 et MS Project 2003

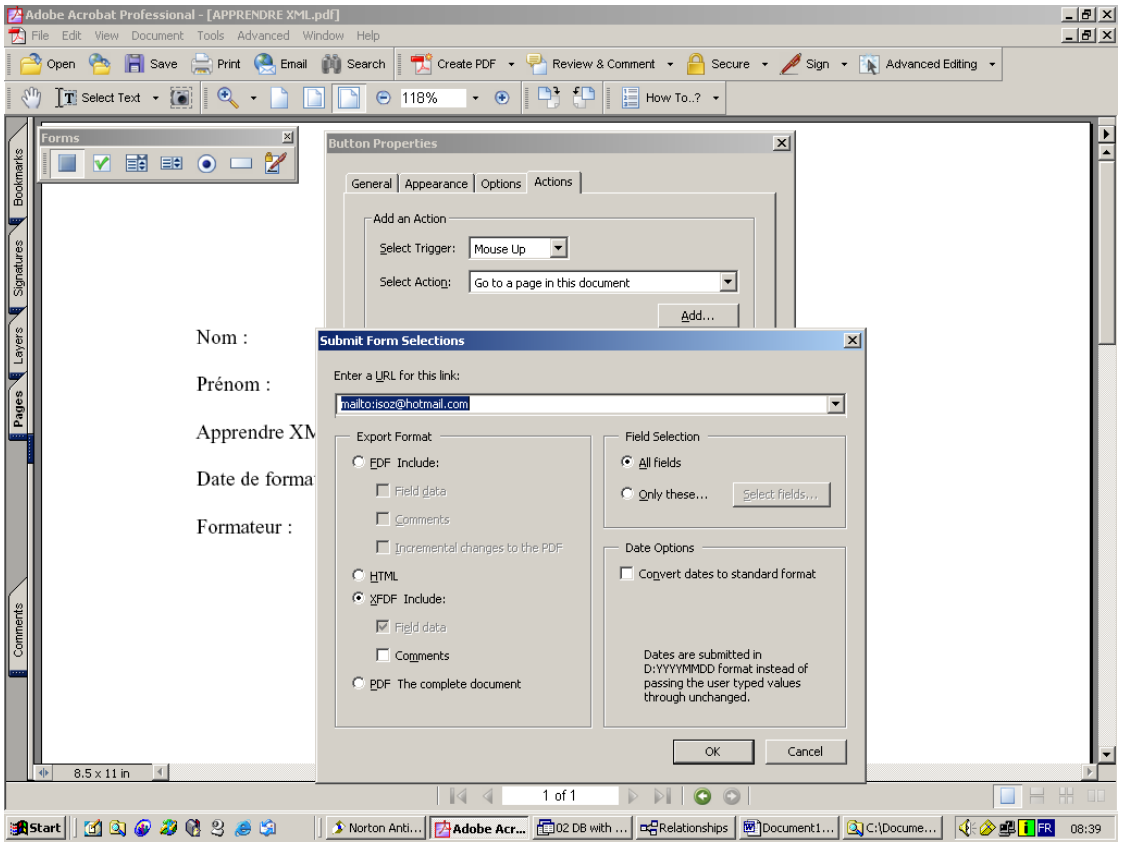
Implication: les données peuvent très facilement être traitées par import XML dans une base MS Access 2003.

Voyons de suite un exemple (à nouveau le but n'est pas d'apprendre dans ce document à créer des formulaires dans Adobe Acrobat mais voir comment faire pour que ceux-ci s'exportent en XML).

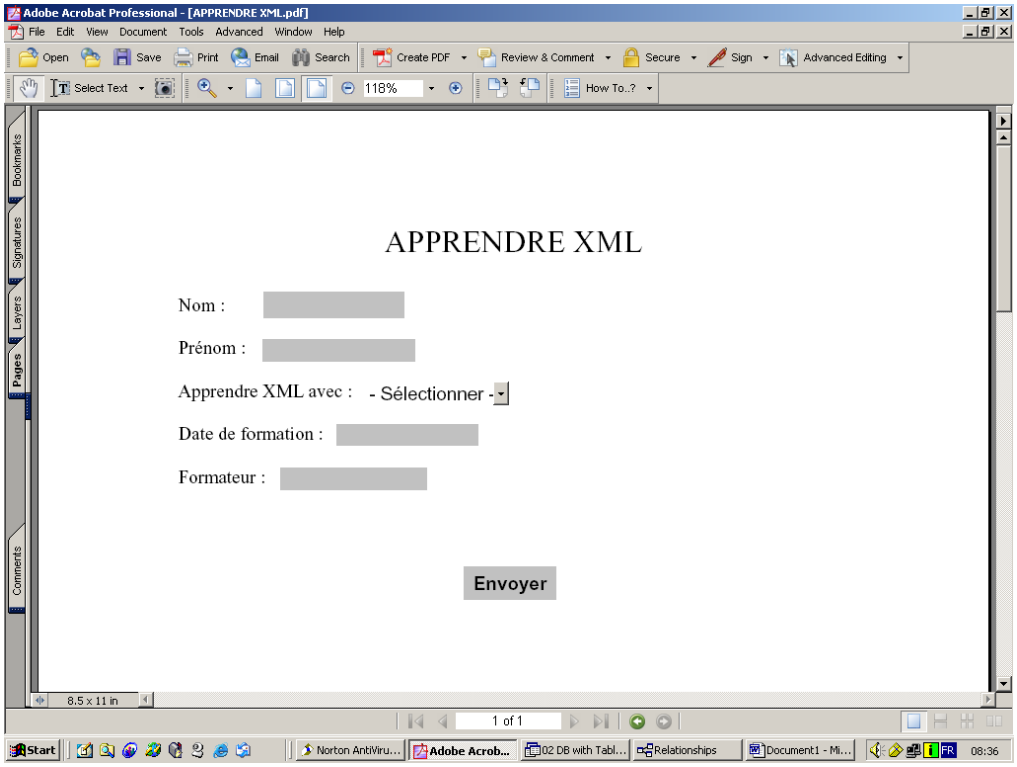
Soit le formulaire suivant sur un thème que nous connaissons bien:



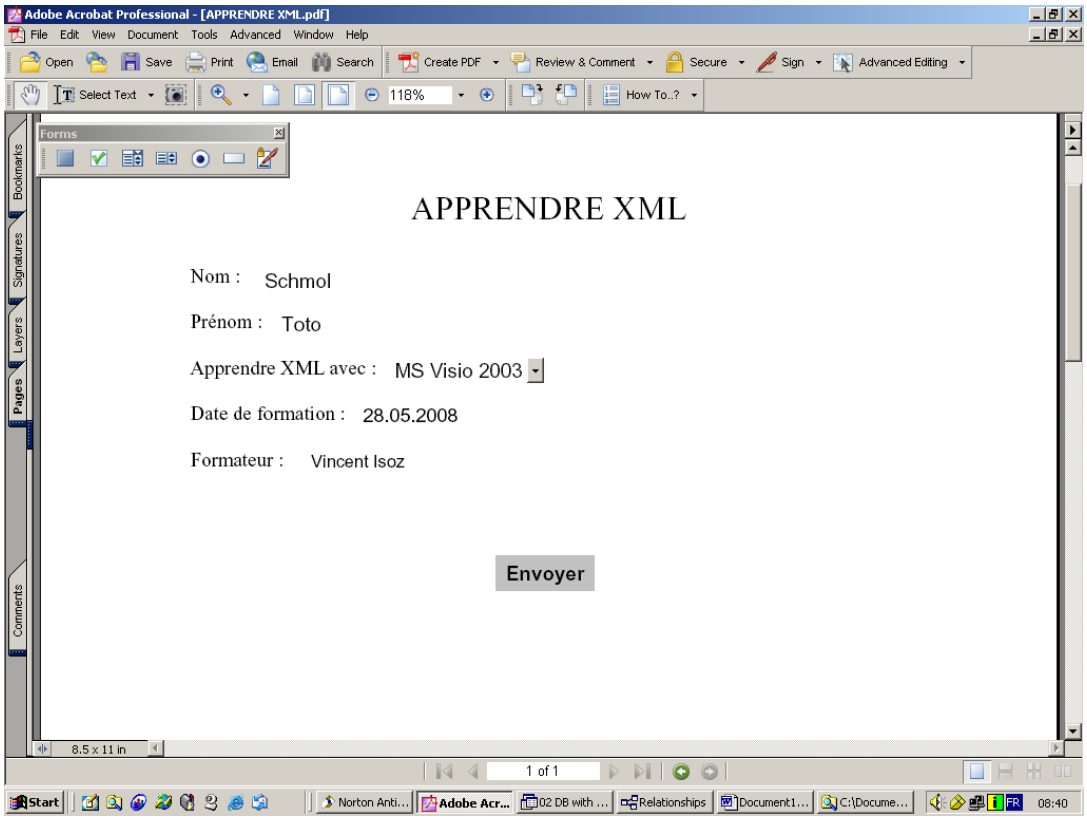
Ajoutons un bouton d'action de validation du formulaire et activons l'option de transformation XFDF (ne pas oublier l'adresse e-mail pour l'envoi des données):



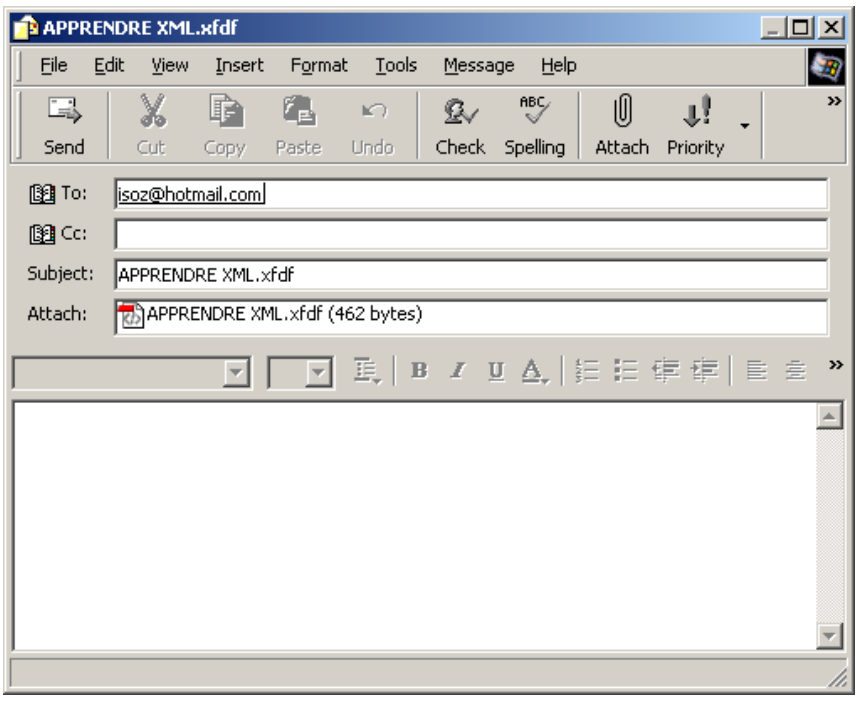
Une fois les options validées, vous obtenez bien un bouton en bas de votre formulaire:



Nous remplissons ce dernier (tiens ! c'est moi le formateur….):



Nous cliquons sur *Envoyer* (un nouveau message électronique s'ouvre avec notre fichier *.xfdf en pièce jointe):



L'utilisateur l'envoie, nous recevons le mail, enregistrons la pièce jointe et la renommons en *.xml et ouvrons ce dernier et que voyons nous:

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
    <f href="/C/Documents and Settings/c6stu1.N0COSA/Desktop/XML/APPRENDRE XML.pdf"/>
```

```
    <ids original="C5F26FE0840A5567CF80842E0A2BA95B"
modified="DA5F7ADA1E5A8E4ABE41909BD4C8DD02"/>
    <fields>
        <field name="Choix">
            <value>MS Visio 2003</value>
        </field>
        <field name="Date">
            <value>28.05.2008</value>
        </field>
        <field name="Envoyer"/>
        <field name="Formateur">
            <value>Vincent ISOZ</value>
        </field>
        <field name="Nom">
            <value>Schmol</value>
        </field>
        <field name="Prenom">
            <value>Toto</value>
        </field>
    </fields>
</xfdf>
```

Nous avons donc un schéma XML parfaitement propre dont chaque balise à comme attribut le nom des champs du formulaire (éviter les accents pour les noms des champs…).

Ensuite, sans se casser la tête, un utilisateur moyen pourra très bien faire une macro (ou un fichier VBScript) qui dans MS Excel 2003 (respectivement depuis MS Windows pour le VBScript) importe les données du formulaires, les transpose (oui malheureusement toutes les données sont au même niveau hiérarchique), exporte les données au format voulu et ensuite on procède de même manuellement (ou automatiquement) dans MS Access 2003 mais pour l'import.

On peut par ailleurs importer ces données dans Adobe Acrobat mais la simplicité de cette action est telle qu'on la voit pendant le cours Adobe Acrobat.

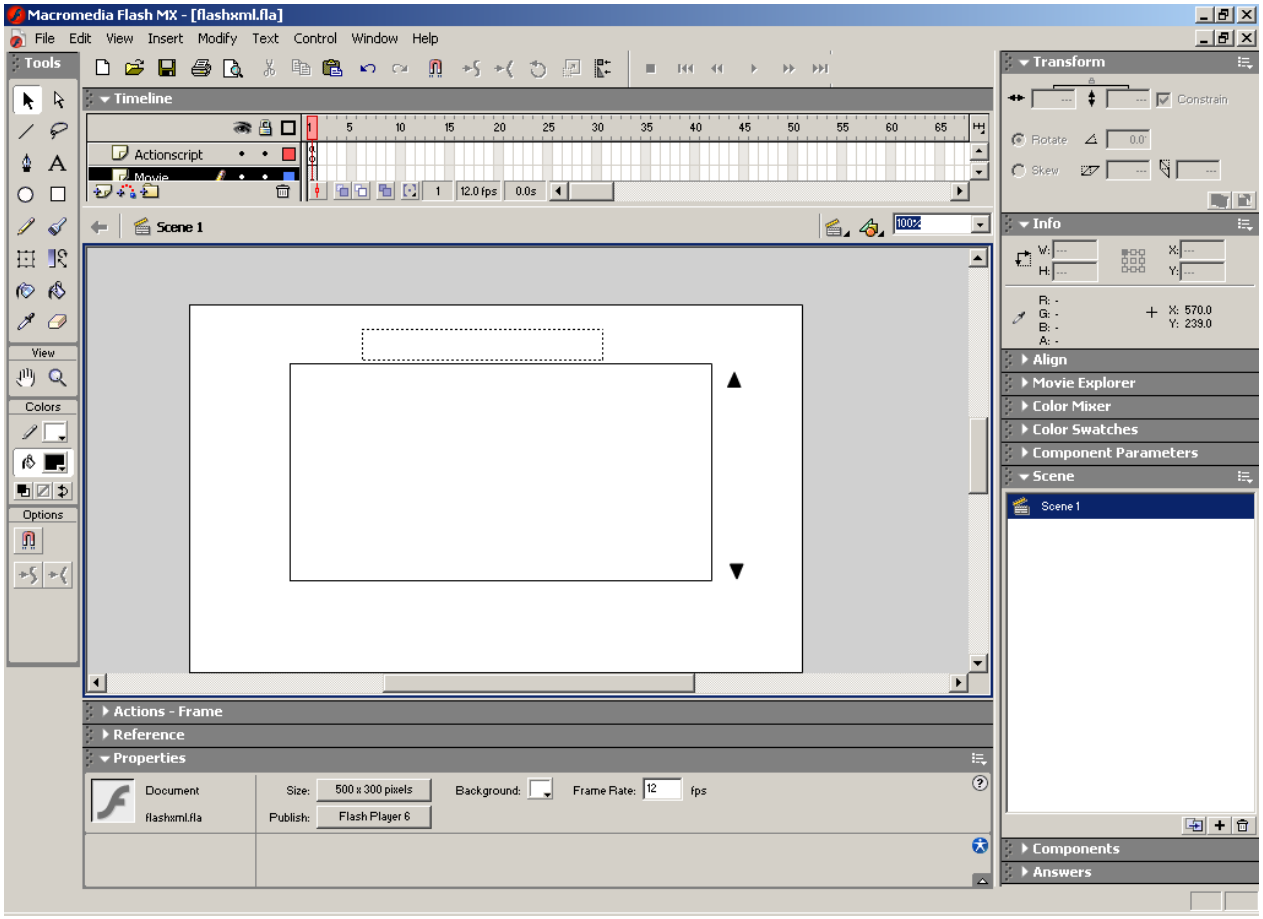
17. XML et FLASH

Nous allons voir maintenant comment transformer un flux RSS dans une textbox Flash.

Voici le fichier XML utilisé (news.xml) d'un site bien connu…. (attention à l'encodage du fichier qui doit être du type UNICODE!!!):

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rss version="2.0">
    <channel>
        <title>Sciences.ch - News</title>
        <link>http://www.sciences.ch</link>
        <description>Sciences News First</description>
        <language>fr</language>
        <copyright>Copyright Sciences.ch</copyright>
        <pubDate>Sun, 20 Apr 2003 08:01:04 PDT</pubDate>
        <lastBuildDate>Sun, 20 Apr 2003 08:01:04 PDT</lastBuildDate>
        <category>Mathématique, Physique théorique</category>
        <generator>Sciences.ch</generator>
        <docs>http://www.sciences.ch/accueil.htm</docs>
        <image>
            <title>Sciences.ch</title>
            <url>http://www.sciences.ch/images/bansciencech.gif</url>
            <link>http://www.sciences.ch</link>
            <width>88</width>
            <height>31</height>
        </image>
        <item>
            <title>Effet Compton relativiste</title>
            <link>http://www.sciences.ch/htmlfr/physatomique/physatomphysnucl01.php#effetcompton</link>
            <description>Dans le domaine de la radioprotection en physique nucleaire, il est important de savoir
comment le rayonnement penetre la matiere et comme celui-ci interagit avec cette derniere. Plusieurs types
d'interactions differentes sont possibles tel que la creation de paires electron-positron, de diffusion coherente ou
incoherente (effet compton), et d'autres phenomenes nucleaires beaucoup plus complexes. Nous proposons de
demontrer au lecteur dans le cadre d'une diffusion incoherent relativiste comment le photon perd de son energie lors de
la collision avec un electron du materiau concerne et quelques proprietes qu'il est possible de tirer de ce modele
theorique.</description>
        </item>
        <item>
            <title>Origine de la chaleur</title>
            <link>http://www.sciences.ch/htmlfr/mecanique/mecanthermodyn01.php#chaleur</link>
            <description>Qu'est-ce que la chaleur ? Voici une question que se posent nombre d'étudiants et parfois
suffisamment longtemps pour qu'ils n'obtiennent jamais la réponse même lors de leur cursus scolaire universitaire. Ne
pas comprendre ce qu'est la chaleur est nous le savons, une source énorme de confusion et de difficulté de
compréhension des conceptions fondamentaux de la thermodynamique (premier principe). Sciences.ch propose donc la
une explication microscopique de la chaleur basée sur la mécanique statistique démontrant que cette première n'est
qu'au fait qu'un changement de la distribution énergétique des micro-états vers de plus hautes ou plus faible valeurs. Il
convient peut-être de se demander s'il ne faudrait pas introduire la thermodynamique seulement et seulement après
avoir étudié les bases de la mécanique statistique  (étant donné les problèmes qu'ont les étudiants en
thermodynamique) qui permettent d'accéder à la démonstration que nous proposons.</description>
            <pubDate>Je, 27 Mai 2004 11:50:01 GMT</pubDate>
        </item>
    </channel>
</rss>
```

Voici à quoi ressemble le fichier news.fla ouvert dans Flash MX:

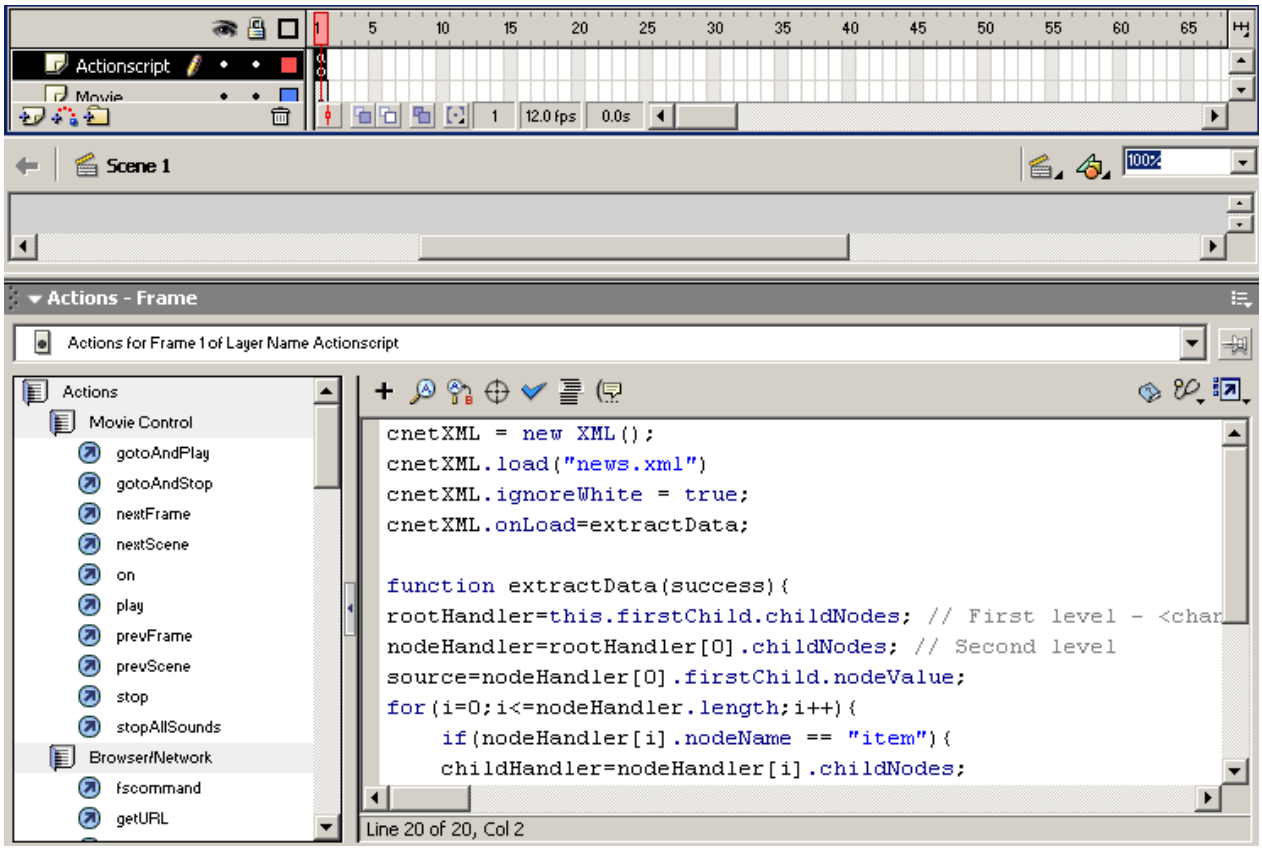


On y voit deux champs texte. Le premier va être rempli avec la balise <title>Sciences.ch - News</title> le second avec les balises:

```
<title>Effet Compton relativiste</title>
<link>http://www.sciences.ch/htmlfr/physatomique/physatomphysnucl01.php#effetcompton</link>
<description>Dans le domaine de la radioprotection en….</description>
```

Pour les remplir nous avons l'actions situé sur la première frame de notre animation:

L'explorateur d'actions nous montre:

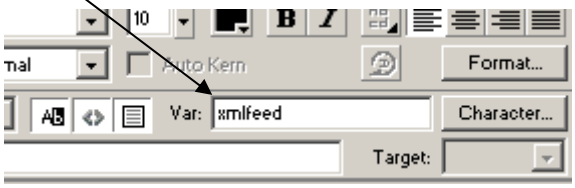


Le code en entier étant:

```
cnetXML = new XML();
cnetXML.load("news.xml")
cnetXML.ignoreWhite = true;
cnetXML.onLoad=extractData;

function extractData(success){
rootHandler=this.firstChild.childNodes; // First level - <channel>
nodeHandler=rootHandler[0].childNodes; // Second level
source=nodeHandler[0].firstChild.nodeValue;
for(i=0;i<=nodeHandler.length;i++){
        if(nodeHandler[i].nodeName == "item"){
        childHandler=nodeHandler[i].childNodes;
        title = childHandler[0].firstChild.nodeValue;
        link = childHandler[1].firstChild.nodeValue;
        description = childHandler[2].firstChild.nodeValue;
        addfeed += "<a href=\""+link+"\" target=\"_new\"><font size=\"+1\"><b>" + title +
"</b></font></a><br>"+description+"<br><br>";
        }
}
xmlfeed=addfeed;
}
```

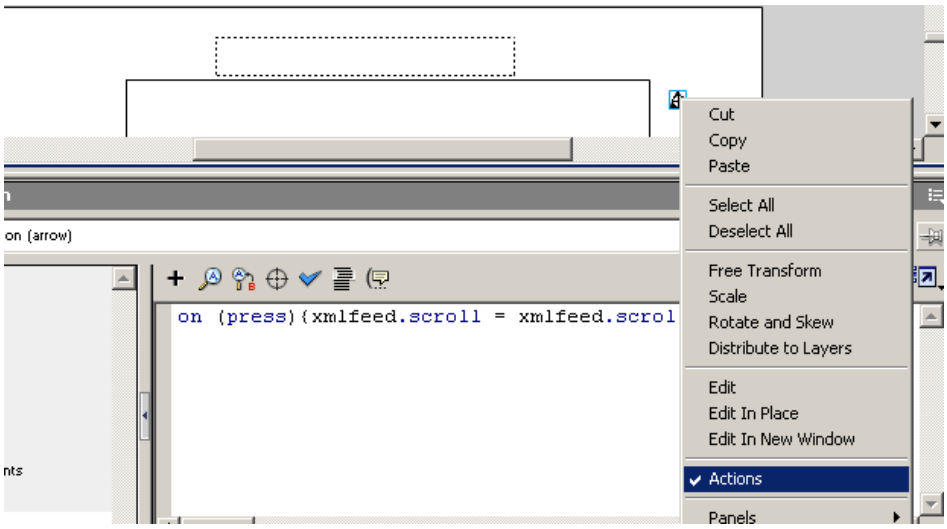
La textbox de titre à alors nécessairement comme nom:

La textbox du flux XML:

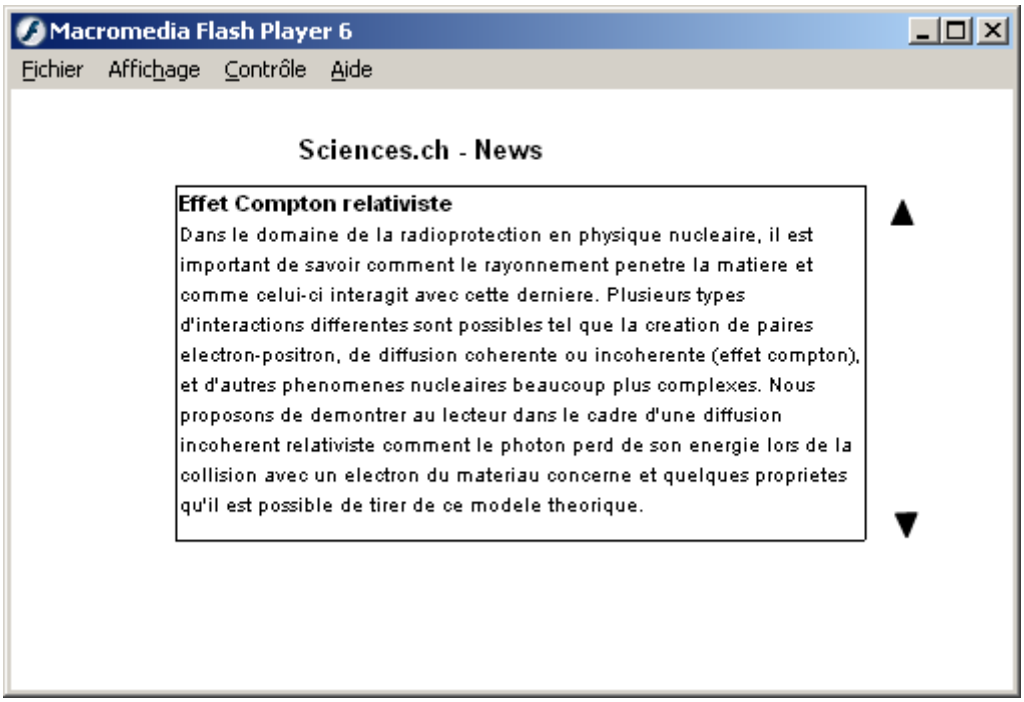


Correspondant bien aux variables que nous trouvons dans l'ActionScript de notre Animation Flash.

Sur les flèches en faisant un clic droit:



Le fichier SWF sortant donne alors:


Macromedia Flash Player 6
Fichier Affichage Contrôle Aide
Sciences.ch - News
Effet Compton relativiste
Dans le domaine de la radioprotection en physique nucleaire, il est important de savoir comment le rayonnement penetre la matiere et comme celui-ci interagit avec cette derniere. Plusieurs types d'interactions differentes sont possibles tel que la creation de paires electron-positron, de diffusion coherente ou incoherente (effet compton), et d'autres phenomenes nucleaires beaucoup plus complexes. Nous proposons de demontrer au lecteur dans le cadre d'une diffusion incoherent relativiste comment le photon perd de son energie lors de la collision avec un electron du materiau concerne et quelques proprietes qu'il est possible de tirer de ce modele theorique.

18. XSL-FO

Les feuilles de style XSL-FO (XSL Formatting Objects) permettent de créer des documents à partir de fichiers XML. Cette norme est le plus souvent utilisée pour générer des fichiers PDF (Acrobat). Il est ainsi possible, dans une feuille de style XSL-FO, de définir la mise en page que devra avoir le document final (en-tête, pied de page, ...), la pagination, ...

Cette norme est complémentaire à XSLT. Alors que XSLT est utilisé pour afficher du contenu XML (dans un navigateur Web par exemple), XSL-FO permettra de transformer ce contenu en un document pouvant être facilement téléchargé et imprimé par l'utilisateur final. Il est ainsi tout à fait possible de proposer sur un site Web le même contenu sous diverses formes sans jamais avoir à toucher au document XML initial.

XSL-FO dérive de plusieurs nomres parmi lesquelles DSSSL (Document Style Sematic and Specification Language) et CSS, et différentes nomres issues de CSS&FP (Cascading Style Sheet & Formatting Property). Dans la mesure où les éléments XSL/FO sont nombreux et complexes, nous nous contenterons d'en étudier quelques'uns pour expliquer le fonctionnement de ce langage.

XSL-FO permet, entre autres, de définir les caractéristiques suivantes d'un document:

- La position des informations
- La définition des bordures et des fonds de page
- La gestion de la césure
- Les caractéristiques des marges et des paragraphes
- L'alignement et la dimension des zones de contenu
- Les polices de caractères, les couleurs

Le traitement d'un document XSL FO se fait en quatre étapes:

1. Parsage du document pour identifier les formatting objects
2. Construction d'un arbre XML correspondant aux éléments, effectuée en plusieurs passes (les éléments FO étant très liés, le processeur ne peut correctement calculer certains d'entre eux sans prendre connaissance des autres)
3. Construction d'un arbre de positionnement (area tree) qui définit le positionnement des éléments sur les différentes pages contenues dans le document correspondant au produit final de la transformation.

Si vous créez un nouveu document XSL-FO avec XML-Spy vous aurez à l'écran:

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="">
            <fo:region-body></fo:region-body>
        </fo:simple-page-master>
```

```
    </fo:layout-master-set>
    <fo:page-sequence master-reference="">
        <fo:flow flow-name="">
            <fo:block></fo:block>
        </fo:flow>
    </fo:page-sequence>

</fo:root>
```

Expliquons le contenu de ce document:

L'élément racine *fo* s'appelle *root*. Celui-ci définit l'URI de l'espace de nom à l'aide de l'attribut xmlns.

Sous la racine figure l'élément *layout-master-set* qui va permettre de définir le type de page produite et leur ordonnancement.

Nous décrivons ensuite le contenu de chaque type de page ou de disposition (layout). Il doit au moins contenir l'élément *region-body* qui permet de décrire le corps d'une page.

Le deuxième élément situé sous la racinese nomme *page-sequence* et possède un attribut *master-reference*. C'est lui qui va indiquer l'ordre des pages utilisées par le document. Les éléments *flow* et *block* en définissent le contenu ainsi que diverses informations de formatage.

18.1 Mise en page

Au début d'une feuille de style XSL-FO, il faut définir les différents types de pages qui composeront le document final. Pour créer un nouveau type de page, il faut employer le modèle suivant:

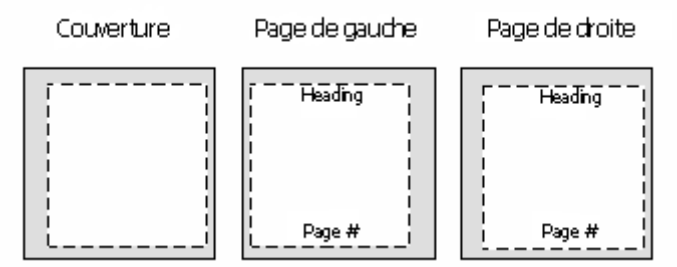
```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="typeName"
            page-height="22cm"
            page-width="14cm"
            margin-top="2cm"
            margin-bottom="2cm"
            margin-left="3cm"
            margin-right="2cm"
            >
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="">
        <fo:flow flow-name="">
            <fo:block></fo:block>
        </fo:flow>
    </fo:page-sequence>

</fo:root>
```

Les attributs suivants sont utilisés:

- *master-name*: nom attribué à ce type de page
- *page-height*: hauteur de la page (en pixels, millimètres, ...)
- *page-width*: largeur de la page
- margin-...: marges du haut, du bas, de gauche et de droite

Dans le cas d'une brochure par exemple, on peut imaginer 3 types de pages:

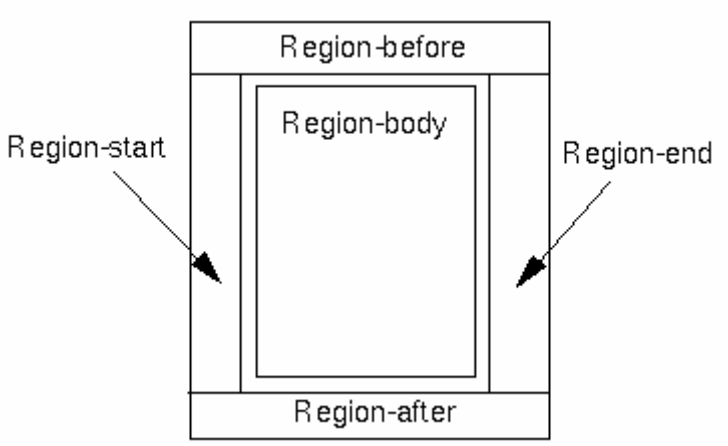
1. Page de couverture
2. Page de gauche - avec une marge à droite pour la reliure
3. Page de droite - avec une marge à gauche pour la reliure



Le contenu de la feuille de style XSL-FO ressemblera à:

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="cover"
            page-height="22cm"
            page-width="14cm"
            margin-top="2cm"
            margin-bottom="2cm"
            margin-left="3cm"
            margin-right="2cm"
        >
        </fo:simple-page-master>
        <fo:simple-page-master master-name="leftPage"
            page-height="22cm"
            page-width="14cm"
            margin-left="2cm"
            margin-right="3cm"
            margin-top="2cm"
            margin-bottom="2cm"
        >
        </fo:simple-page-master>
        <fo:simple-page-master master-name="rightPage"
            page-height="22cm"
            page-width="14cm"
            margin-left="3cm"
            margin-right="2cm"
            margin-top="2cm"
            margin-bottom="2cm"
        >
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="">
        <fo:flow flow-name="">
            <fo:block></fo:block>
        </fo:flow>
    </fo:page-sequence>

</fo:root>
```

Après avoir défini des modèles, il faut paramétrer les diverses zones de contenu qui figureront dans les pages. FOP reconnaît 5 zones de contenus différentes:



region-body	*region-before*	*egion-after*	*region-start*	*region-end*
correspond au corps de texte	correspond à un en-tête de page	correspond à un pied de page	correspond à une zone située à gauche du body	correspond à une zone située à droite du body

Il faut savoir que XSL-FO a été prévu pour travailler avec tous types de caractères. Il est ainsi tout à fait possible d'écrire du contenu de gauche à droite (comme en français), de droite à gauche (comme l'arabe) ou encore de haut en bas (comme le Japonais dans certains cas).

Les pages de gauche et de droite nécessitent un en-tête et un pied de page, ce qui n'est pas le cas de la page de garde. Cette dernière n'a besoin que d'un body.

La feuille de style XSL-FO avec les paramétrages des régions est décrite ci-dessous. Les en-têtes et les pieds de page des pages gauche et droite font 1 cm de hauteur. Quand au body, il se situe à 1.1cm du haut et du bas de la page, ce qui laisse une marge de 0.1cm entre l'en-tête et le pied de page et le body (enregistrez ce code sous le nom *Test.fo*):

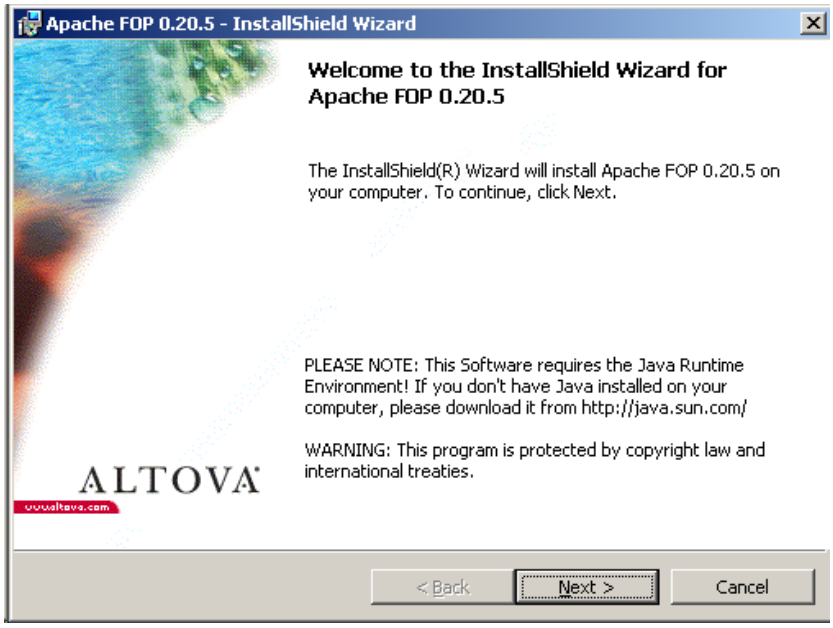
```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="cover"
            page-height="22cm"
            page-width="14cm"
            margin-top="2cm"
            margin-bottom="2cm"
            margin-left="3cm"
            margin-right="2cm"
        >

            <fo:region-body margin-top="4cm" />


        </fo:simple-page-master>
        <fo:simple-page-master master-name="leftPage"
            page-height="22cm"
            page-width="14cm"
            margin-left="2cm"
            margin-right="3cm"
            margin-top="2cm"
            margin-bottom="2cm"
        >

            <fo:region-before extent="1cm"/>
            <fo:region-after extent="1cm"/>
            <fo:region-body margin-top="1.1cm" margin-bottom="1.1cm" />
```

```
        </fo:simple-page-master>
        <fo:simple-page-master master-name="rightPage"
            page-height="22cm"
            page-width="14cm"
            margin-left="3cm"
            margin-right="2cm"
            margin-top="2cm"
            margin-bottom="2cm"
        >

            <fo:region-before extent="1cm"/>
            <fo:region-after extent="1cm"/>
            <fo:region-body margin-top="1.1cm" margin-bottom="1.1cm" />

        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="cover">
        <fo:flow flow-name="">
            <fo:block></fo:block>
        </fo:flow>
    </fo:page-sequence>
        <fo:page-sequence master-reference="leftPage">
        <fo:flow flow-name="">
            <fo:block></fo:block>
        </fo:flow>
    </fo:page-sequence>
        <fo:page-sequence master-reference="rightPage">
        <fo:flow flow-name="">
            <fo:block></fo:block>
        </fo:flow>
    </fo:page-sequence>

</fo:root>
```

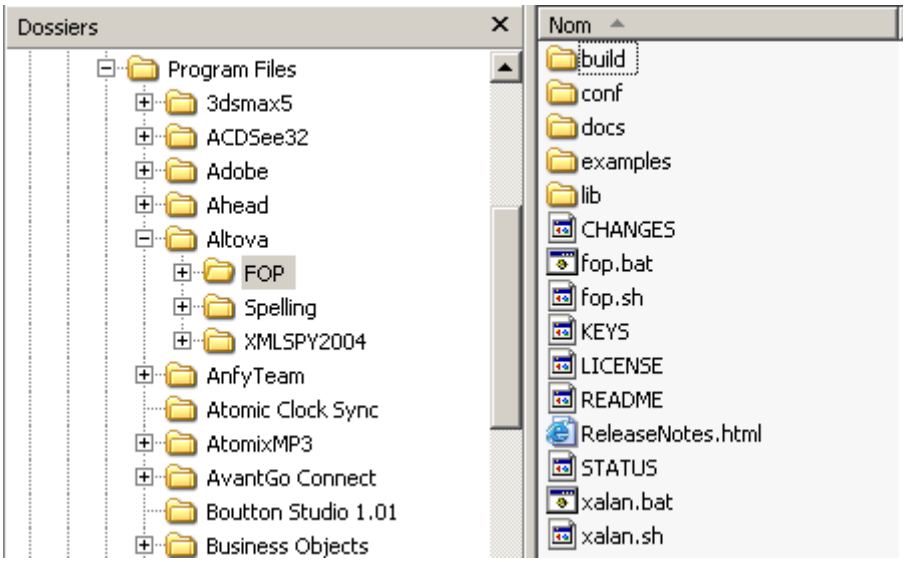
Nous pouvons déjà compiler l'exemple ci-dessous. Pour ce faire, il vous faudra télécharger ApacheFOP à l'adresse suivante:

http://link.altova.com/download/2005/ApacheFOP.exe

et l'installer en bonne et due forme:



qui sera installé dans le répertoire par défaut:



ensuite, copier le fichier *Test.of* dans le dossier *FOP* ci-dessous. Ensuite, ouvrez une fenêtre prompt:

```
C:\>cd \program files\altova\fop

C:\Program Files\Altova\FOP>_
```

et tapez:

```
C:\Program Files\Altova\FOP>fop.bat test.fo test.pdf
[INFO] Using org.apache.xerces.parsers.SAXParser as SAX2 Parser
[INFO] FOP 0.20.5
[INFO] Using org.apache.xerces.parsers.SAXParser as SAX2 Parser
[INFO] building formatting object tree
[INFO] setting up fonts
[WARNING] A 'flow-name' is required for fo:flow. This constraint will be enforce
d in future versions of FOP
[INFO] [1]
[WARNING] A 'flow-name' is required for fo:flow. This constraint will be enforce
d in future versions of FOP
[INFO] [2]
[WARNING] A 'flow-name' is required for fo:flow. This constraint will be enforce
d in future versions of FOP
[INFO] [3]
[INFO] Parsing of document complete, stopping renderer
```

Vous avez ensuite votre fichier PDF… suffit de l'ouvrir ;-)

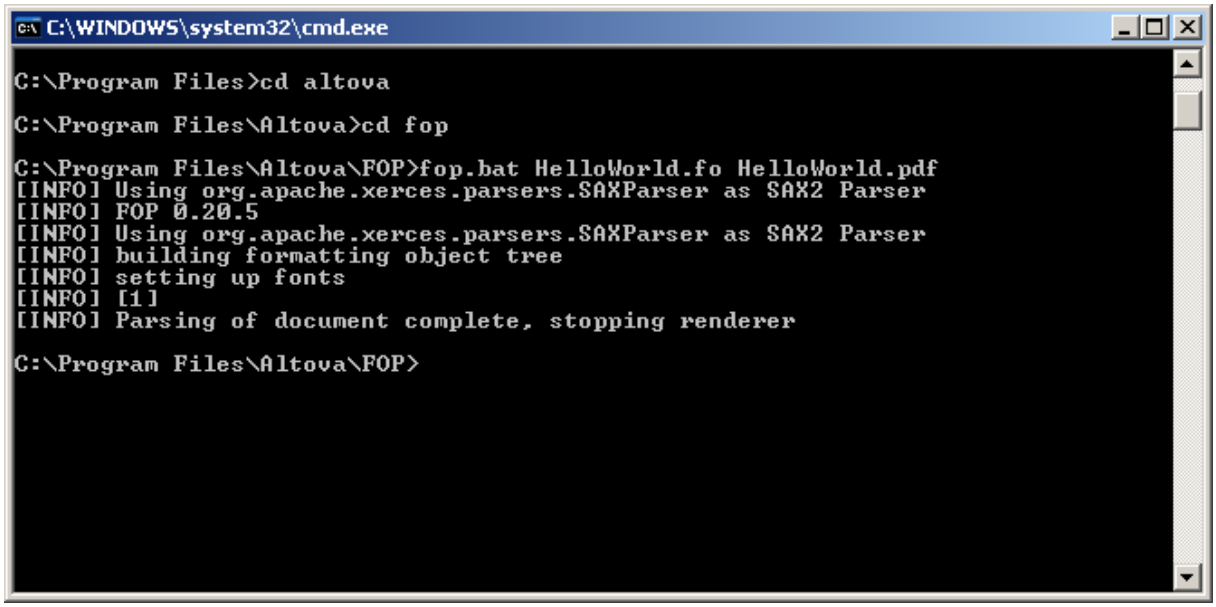
18.1.1 Création fichier PDF "Hello World"

Maintenant dans XMLSpy créez un fichier *.fo que nous enregistrerons sous le nom HelloWorld.fo dans le dossier *Altova/FOP* tel qu'il contienne:

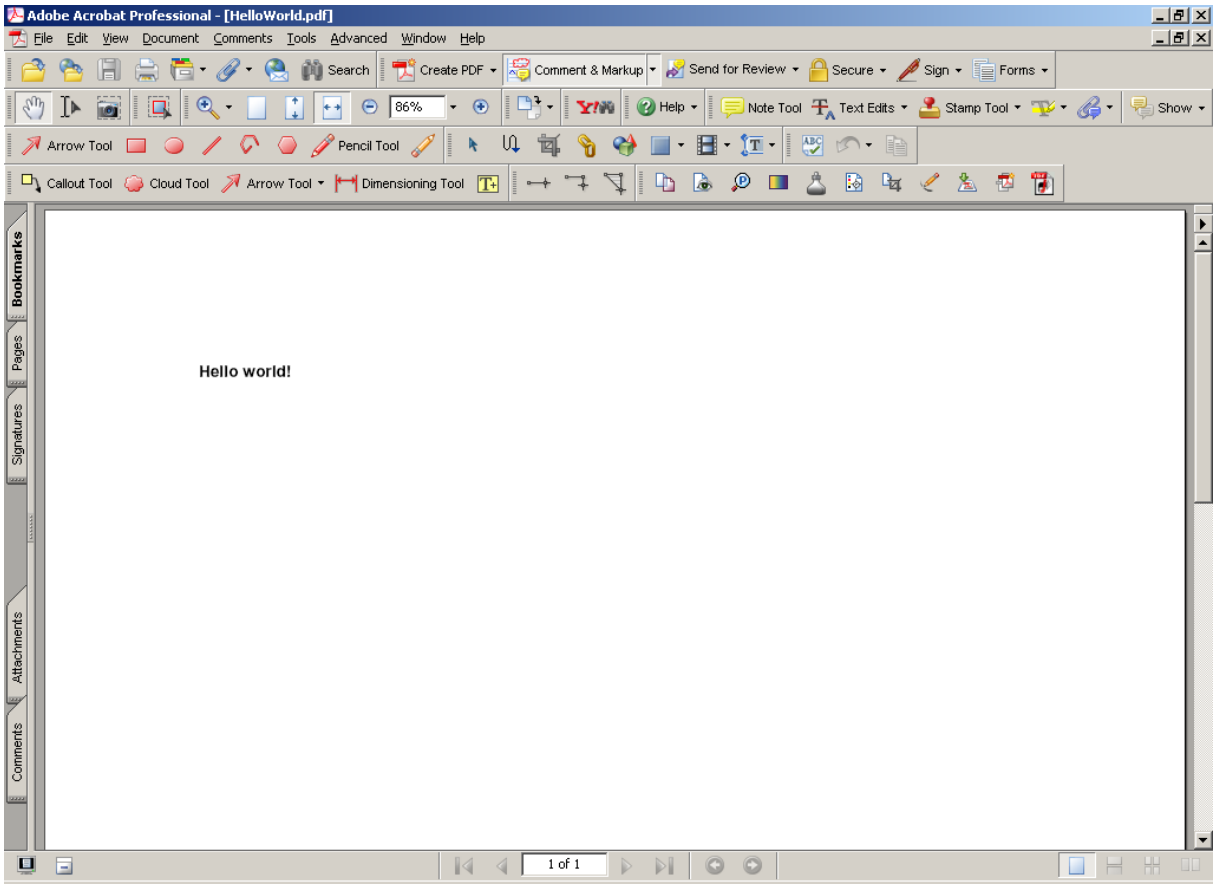
```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="A4" page-width="297mm" page-height="210mm" margin-top="1cm"
margin-bottom="1cm" margin-left="1cm" margin-right="1cm">
            <fo:region-body margin="3cm"/>
            <fo:region-before extent="2cm"/>
            <fo:region-after extent="2cm"/>
            <fo:region-start extent="2cm"/>
            <fo:region-end extent="2cm"/>
        </fo:simple-page-master>
    </fo:layout-master-set>
```

```
    <fo:page-sequence master-reference="A4" format="A">
        <fo:flow flow-name="xsl-region-body">
            <fo:block>
                <fo:inline font-weight="bold">Hello world!</fo:inline>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>

</fo:root>
```

Ensuite, en ligne de commande, vous écrivez:



Le résultat étant un fichier PDF à la sortie dans le même dossier que là ou se situe votre fichier *.fo original:

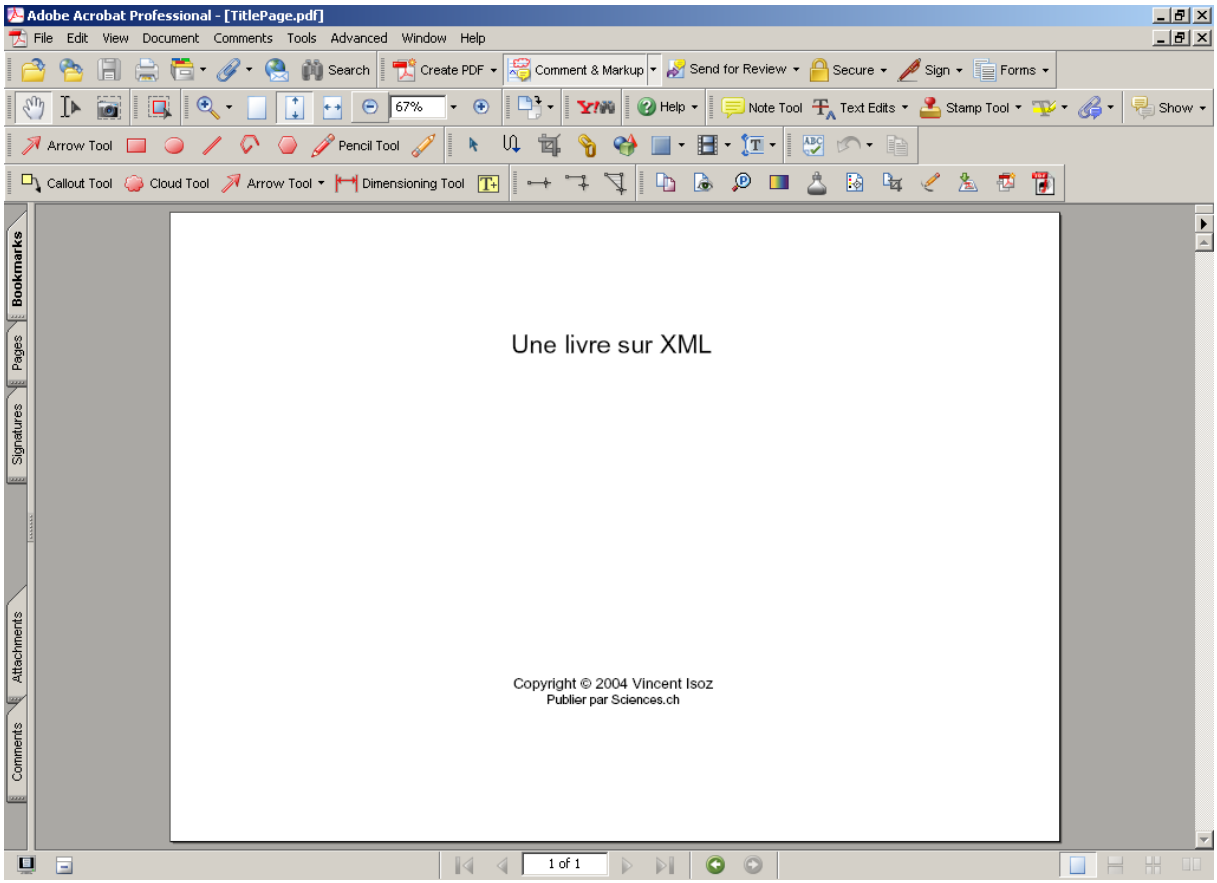


18.1.2 Page de titre

Encore un autre exemple à enregistrer sour le nom *TitlePage.fo*:

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="cover" page-width="297mm" page-height="210mm" margin-top="1cm"
margin-bottom="1cm" margin-left="1cm" margin-right="1cm">
            <fo:region-body margin="3cm"/>
            <fo:region-before extent="2cm"/>
            <fo:region-after extent="2cm"/>
            <fo:region-start extent="2cm"/>
            <fo:region-end extent="2cm"/>
        </fo:simple-page-master>
    </fo:layout-master-set>

<fo:page-sequence master-reference="cover">
    <fo:flow flow-name="xsl-region-body">
        <fo:block font-size="24pt" text-align="center" space-after="300pt">
            Une livre sur XML
        </fo:block>
        <fo:block font-size="14pt" text-align="center">
            Copyright &#169; 2004 Vincent ISOZ
            </fo:block>
        <fo:block text-align="center">
            Publié par Sciences.ch
        </fo:block>
    </fo:flow>
</fo:page-sequence>

</fo:root>
```

qui donnera après transformation en ligne de commande:



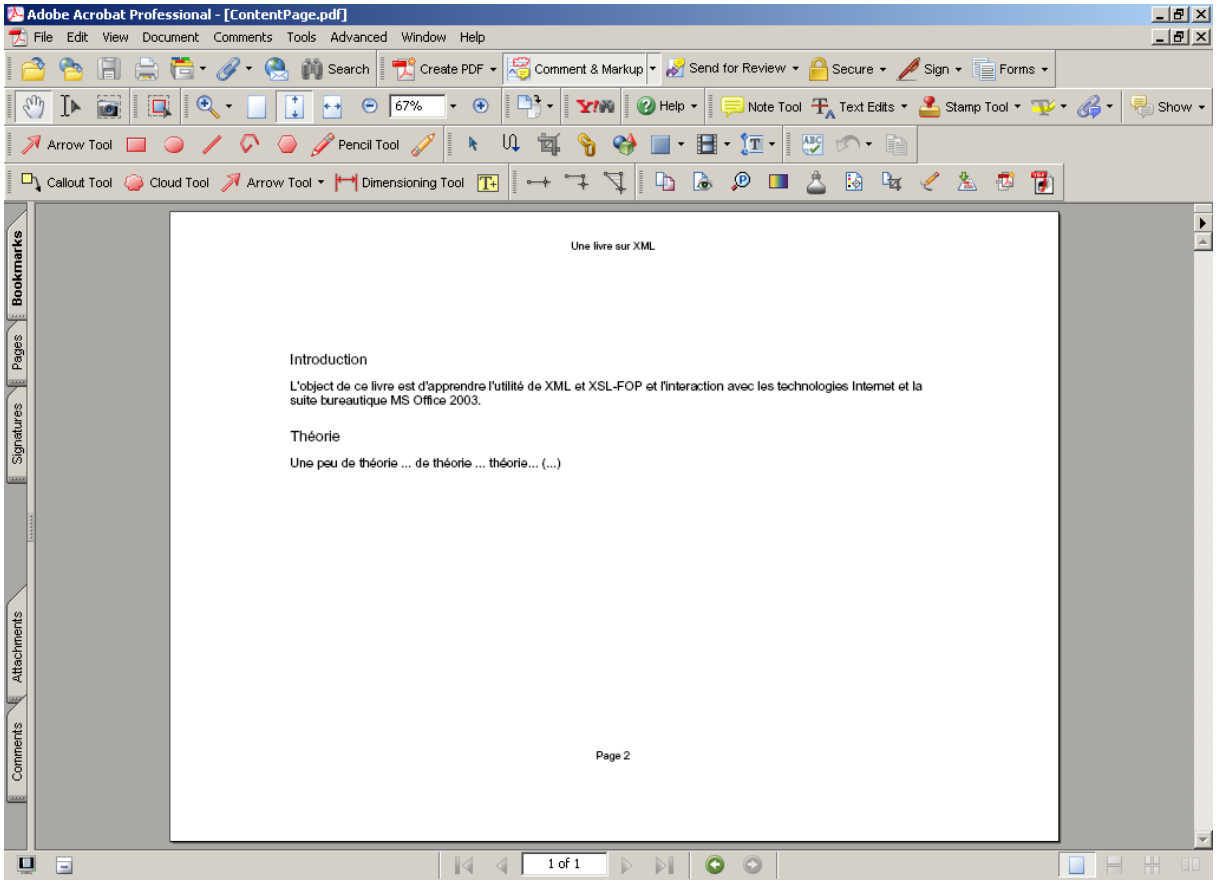
18.1.3 Contenu

Encre un autre exemple un peu plus compliqué maintenant à enregistrer sous le nom *ContentPage.fo*:

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="contents" page-width="297mm" page-height="210mm" margin-
top="1cm" margin-bottom="1cm" margin-left="1cm" margin-right="1cm">
            <fo:region-body margin="3cm"/>
            <fo:region-before extent="2cm"/>
            <fo:region-after extent="2cm"/>
            <fo:region-start extent="2cm"/>
            <fo:region-end extent="2cm"/>
        </fo:simple-page-master>
    </fo:layout-master-set>

<fo:page-sequence master-reference="contents" initial-page-number="2">>
    <fo:static-content flow-name="xsl-region-before">
        <fo:block font-family="Helvetica" font-size="10pt" text-align="center">
            Une livre sur XML
        </fo:block>
    </fo:static-content>
    <fo:static-content flow-name="xsl-region-after">
        <fo:block font-family="Helvetica" font-size="10pt" text-align="center">
            Page <fo:page-number />
        </fo:block>
    </fo:static-content>
    <fo:flow flow-name="xsl-region-body">
        <fo:block font-size="14pt" space-before="20pt" space-after="10pt">
            Introduction
        </fo:block>
         <fo:block font-size="12pt">
```

```
            L'object de ce livre est d'apprendre l'utilité de XML et XSL-FOP et l'interaction avec les technologies Internet et la suite bureautique MS Office 2003.
        </fo:block>
        <fo:block font-size="14pt" space-before="20pt" space-after="10pt">
            Théorie
        </fo:block>
        <fo:block font-size="12pt">
            Une peu de théorie ... de théorie ... théorie... (...)
        </fo:block>
    </fo:flow>
</fo:page-sequence>

</fo:root>
```

Ce qui donnera après compilation:



18.1.4 Pages multiples

Rédigez le code XSL-FO suivant dans un fichier nommé *bibliothque.fo*:

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>

        <fo:simple-page-master master-name="couverture" page-height="10cm" page-width="10cm" margin-left="0.5cm" margin-right="0.5cm" margin-top="0.5cm" margin-bottom="0.5cm">

            <!--pour les en-têtes et pieds de page de la page de couverture-->
            <fo:region-before extent="2.5cm"/>
            <fo:region-after extent="2.5cm"/>
            <fo:region-body margin-top="2.8cm" margin-bottom="2.8cm"/>

        </fo:simple-page-master>
```

```
        <fo:simple-page-master master-name="page interieure gauche" page-height="10cm" page-width="10cm"
margin-left="1cm" margin-right="0.5cm" margin-top="0.5cm" margin-bottom="0.5cm">

            <!--pour les en-têtes et pieds de page de la page interieure gauche-->
            <fo:region-before extent="2.5cm"/>
            <fo:region-after extent="2.5cm"/>
            <fo:region-body margin-top="2.8cm" margin-bottom="2.8cm"/>

        </fo:simple-page-master>

        <fo:simple-page-master master-name="page interieure droite" page-height="10cm" page-width="10cm" margin-
left="0.5cm" margin-right="1cm" margin-top="0.5cm" margin-bottom="0.5cm">

            <!--pour les en-têtes et pieds de page interieure droite-->
            <fo:region-before extent="2.5cm"/>
            <fo:region-after extent="2.5cm"/>
            <fo:region-body margin-top="2.8cm" margin-bottom="2.8cm"/>

        </fo:simple-page-master>

            <!--pour alterner les pages paires ou impaires-->
        <fo:page-sequence-master master-name="contenu">
            <fo:repeatable-page-master-alternatives>
                <fo:conditional-page-master-reference master-reference="page interieure gauche" odd-or-
even="even"/>
                <fo:conditional-page-master-reference master-reference="page interieure droite" odd-or-even="odd"/>
            </fo:repeatable-page-master-alternatives>
        </fo:page-sequence-master>

    </fo:layout-master-set>

    <fo:page-sequence master-reference="couverture">
            <fo:flow flow-name="xsl-region-body">
            <fo:block font-family="Helvetica" font-size="22pt" text-align="start">Ma bibliotheque</fo:block>
            <fo:block font-family="Helvetica" font-size="15pt" text-align="start">Copyright Sciences.ch</fo:block>
            <fo:block text-align="end">Des supports de cours qui sont bien</fo:block>
            <fo:block font-size="10pt" text-align="end">et un bien chouette exemple !</fo:block>
        </fo:flow>
    </fo:page-sequence>


    <fo:page-sequence master-reference="contenu" initial-page-number="1">
        <fo:static-content flow-name="xsl-region-before"      >
            <fo:block font-family="Helvetica" font-size="12pt" text-align="center">Mes supports</fo:block>
        </fo:static-content>
        <fo:static-content flow-name="xsl-region-after">
            <fo:block font-family="Helvetica" font-size="12pt" text-align="center">Copyright Sciences.ch</fo:block>
        </fo:static-content>
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="12pt">Ici on décrit un premier album</fo:block>
            <fo:block font-size="14pt">Par exemple "Le combat ordinaire"</fo:block>
        </fo:flow>
    </fo:page-sequence>

    <fo:page-sequence master-reference="contenu" initial-page-number="2">
        <fo:static-content flow-name="xsl-region-before"      >
            <fo:block font-family="Helvetica" font-size="6pt" text-align="center">Mes supports</fo:block>
        </fo:static-content>
        <fo:static-content flow-name="xsl-region-after">
            <fo:block font-family="Helvetica" font-size="6pt" text-align="center">Page <fo:page-number/></fo:block>
        </fo:static-content>
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="12pt">Ici on décrit un deuxième album</fo:block>
            <fo:block font-size="14pt">Par exemple "Le tueur"</fo:block>
        </fo:flow>
    </fo:page-sequence>


</fo:root>
```

Vous obtiendrez après compilation:

Remarque: notez que la parseur ajoute une page blanche par défaut (comme chez Latex humm…).

18.1.5 XML et XSL-FO

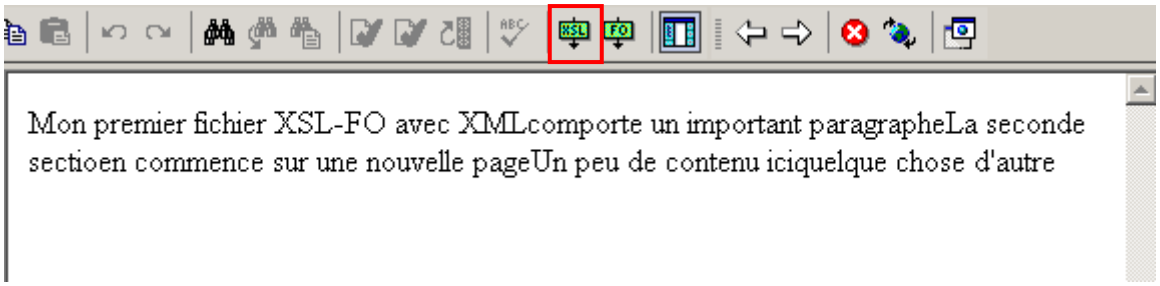
Dans ce dernier exemple, nous allons combiner la puissance du XSL-FO avec un fichier XML. Ecrivez le code suivant dans un fichier nommé *Texte.xml*:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="FO.xsl"?>
<document>
    <section>
        <head>Mon premier fichier XSL-FO avec XML</head>
        <para>comporte un <em>important</em>  paragraphe</para>
    </section>
    <section>
        <head>La seconde section commence sur une nouvelle page</head>
        <para>Un peu de contenu ici</para>
        <foreign>quelque chose d'autre</foreign>
    </section>

</document>
```

Ensuite, créez un fichier XSL nommé *FO.xsl* avec le code suivant:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:fo="http://www.w3.org/1999/XSL/Format"
version="1.0">
  <xsl:output method="xml"/>
  <xsl:template match="/">

       <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
          <fo:layout-master-set>
               <fo:simple-page-master  master-name="simple"
                          page-height  ="29.7cm"
                          page-width   ="21cm"
                          margin-left  ="2.5cm"
                     margin-right ="2.5cm">
                <fo:region-body margin-top="3cm"/>
          </fo:simple-page-master>
         </fo:layout-master-set>

          <fo:page-sequence master-reference="simple">
            <fo:flow flow-name="xsl-region-body">
                <xsl:apply-templates/>
            </fo:flow>
          </fo:page-sequence>
     </fo:root>

  </xsl:template>

  <xsl:template match="document">
     <fo:block>
          <xsl:apply-templates/>
     </fo:block>
  </xsl:template>


<xsl:template match="section">
<fo:block break-before="page">
  <xsl:apply-templates/>
</fo:block>
</xsl:template>
 <xsl:template match="head">
   <fo:block font-size="14pt" font-weight="bold" space-after="1cm" space-after.conditionality = 'retain'>
     <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

  <xsl:template match="para">
    <fo:block>
     <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

  <xsl:template match="em">
   <fo:inline font-style="italic">
     <xsl:apply-templates/>
   </fo:inline>
  </xsl:template>

<xsl:template match="*">
  <fo:block background-color="red">
   <xsl:apply-templates/>
  </fo:block>
 </xsl:template>


</xsl:stylesheet>
```

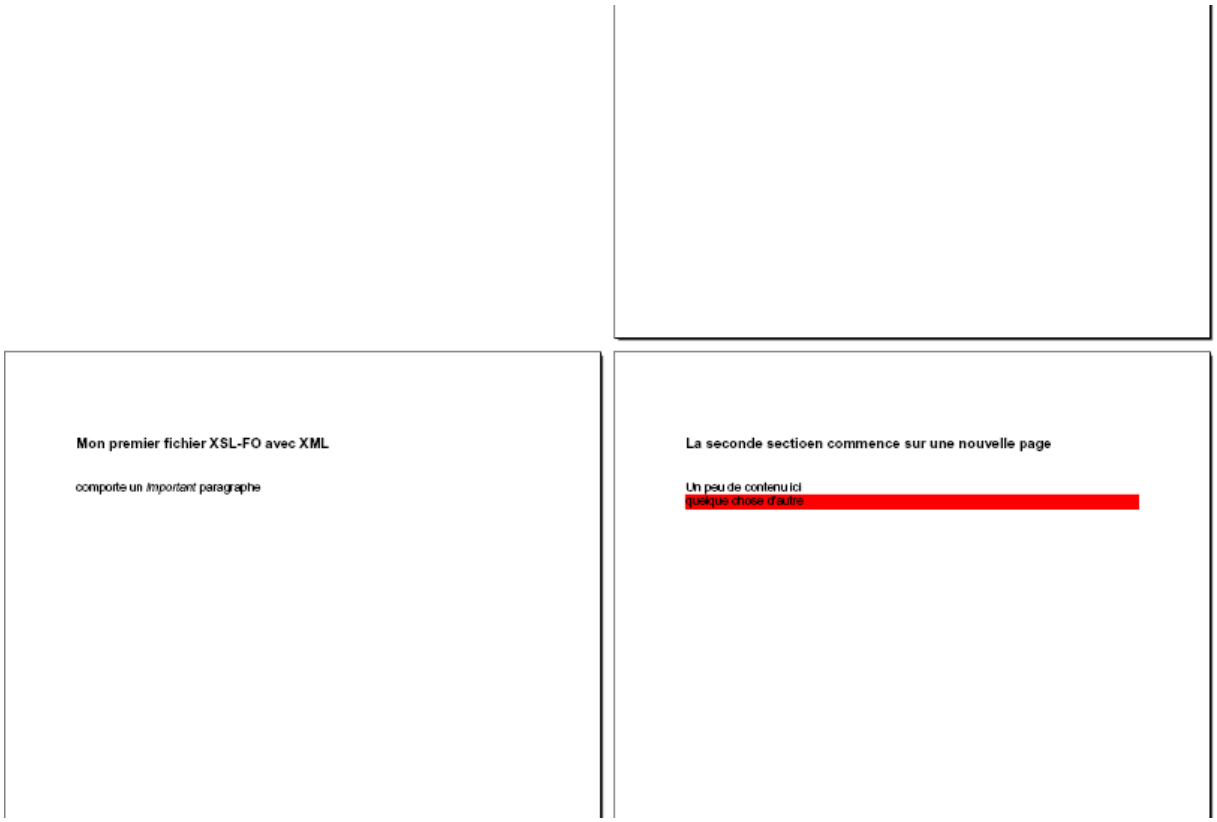
Ensuite, dans XMLSpy allez dans le fichier XML créé précédemment et cliquez sur:



Le code source du fichier sortant est:

```
<?xml version="1.0" encoding="UTF-8"?><fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"><fo:layout-master-set><fo:simple-page-master master-name="simple" page-height="29.7cm" page-width="21cm" margin-left="2.5cm" margin-right="2.5cm"><fo:region-body margin-top="3cm"/></fo:simple-page-master></fo:layout-master-set><fo:page-sequence master-reference="simple"><fo:flow flow-name="xsl-region-body"><fo:block><fo:block break-before="page"><fo:block font-size="14pt" font-weight="bold" space-after="1cm" space-after.conditionality="retain">Mon premier fichier XSL-FO avec XML</fo:block><fo:block>comporte un <fo:inline font-style="italic">important</fo:inline> paragraphe</fo:block></fo:block><fo:block break-before="page"><fo:block font-size="14pt" font-weight="bold" space-after="1cm" space-after.conditionality="retain">La seconde sectioen commence sur une nouvelle page</fo:block><fo:block>Un peu de contenu ici</fo:block><fo:block background-color="red">quelque chose d'autre</fo:block></fo:block></fo:block></fo:flow></fo:page-sequence></fo:root>
```

qui par copier/coller dans un fichier *.fo et par compilation par FOP donne le fichier PDF ci-dessous:



19. MATHML

Le langage MathML est relativement lourd, indigeste et peu esthétique mais il à pour objectif de standardiser les écritures des équations mathématiques sur le web. C'est une forme de standardisation de ce que Latex ou MathType proposent à ce jour.

Pour le faire fonctionner, les utilisateurs doivent à ce jour télécharger le plug-in à l'adresse ci-dessous:

http://www.dessci.com/en/products/mathplayer/

Voyons un exemple de MathML:

```
<m:math style='background-color:#'>
<m:mrow>
<m:mfrac><m:mrow><m:msup><m:mi>x</m:mi><m:mn>2</m:mn></m:msup>
</m:mrow><m:mrow><m:msup><m:mi>a</m:mi>
<m:mn>2</m:mn></m:msup></m:mrow> </m:mfrac>
<m:mo>+</m:mo>
<m:mfrac><m:mrow><m:msup><m:mi>y</m:mi><m:mn>2</m:mn></m:msup></m:mrow>
<m:mrow>
<m:msup><m:mi>b</m:mi><m:mn>2</m:mn></m:msup></m:mrow></m:mfrac>
<m:mo>=</m:mo><m:mn>1</m:mn>
</m:mrow>
</m:math>
```

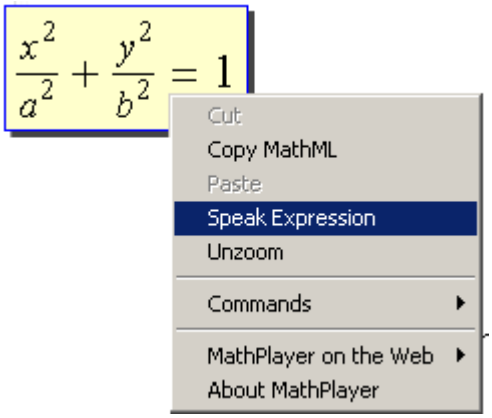
Pour obtenir à l'écran:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

qui quand nous cliquons dessus fait apparaît un zoom:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

et suite à un bouton droit dessus dans MS Windows XP ou ultérieur permet aux aveugles d'entre l'équation:

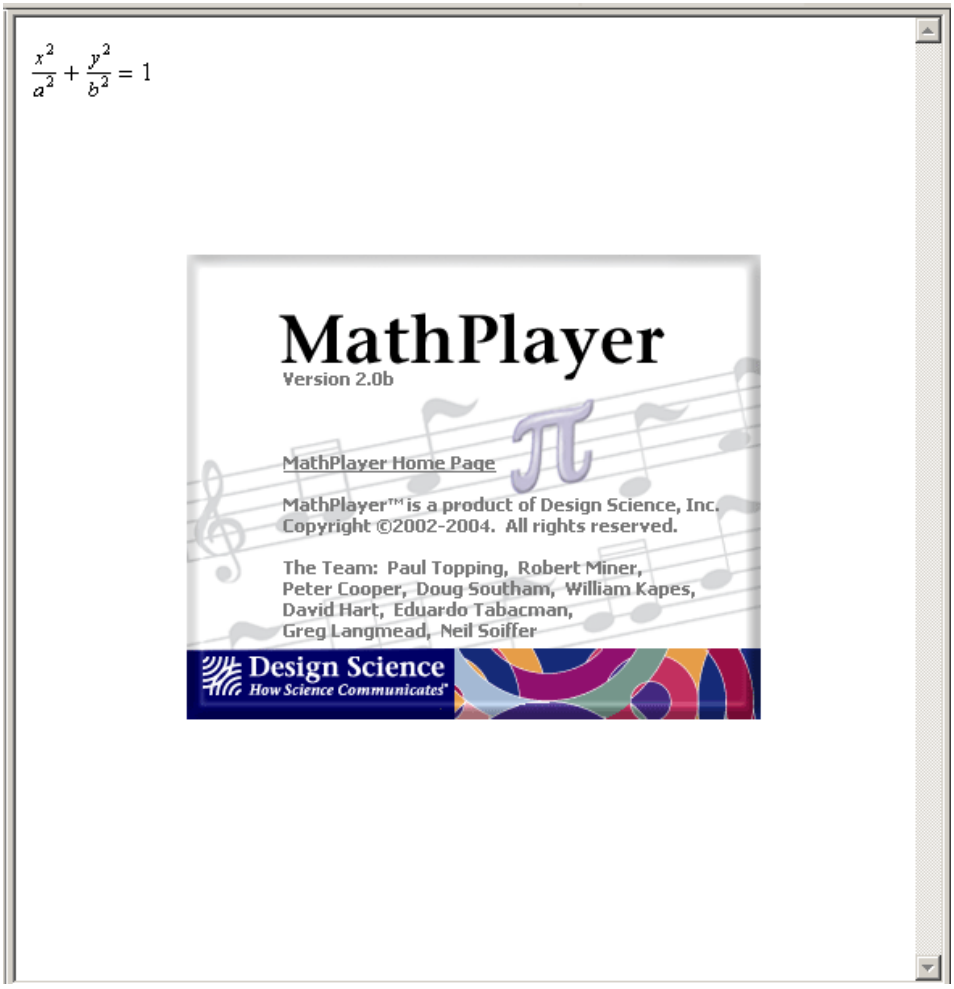


Il existe de nombreux logiciels pour utiliser MathML sans avoir à apprendre la langage. Citons pour les systèmes MS Windows: MathType.

Voici le contexte complet du code MathML relativement à l'exemple précédent:

```
<html xmlns:v="urn:schemas-microsoft-com:vml"
xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:w="urn:schemas-microsoft-com:office:word"
xmlns:dt="uuid:C2F41010-65B3-11d1-A29F-00AA00C14882"
xmlns="http://www.w3.org/TR/REC-html40"
xmlns:m="http://www.w3.org/1998/Math/MathML">
<head>
<object id="MathPlayer" codebase="http://www.dessci.com/dl/mathplayer.cab" classid="clsid:32F66A20-7614-11D4-
BD11-00104BD3F987"></object>
<import namespace='m' implementation='#MathPlayer'>
</head>
<body>
<p class=MsoNormal>
<m:math style='background-color:#'>
<m:mrow>
<m:mfrac><m:mrow><m:msup><m:mi>x</m:mi><m:mn>2</m:mn></m:msup>
</m:mrow><m:mrow><m:msup><m:mi>a</m:mi> <m:mn>2</m:mn></m:msup></m:mrow> </m:mfrac>
<m:mo>+</m:mo>
<m:mfrac><m:mrow><m:msup><m:mi>y</m:mi><m:mn>2</m:mn></m:msup></m:mrow>
<m:mrow> <m:msup><m:mi>b</m:mi><m:mn>2</m:mn></m:msup></m:mrow></m:mfrac>
<m:mo>=</m:mo><m:mn>1</m:mn>
</m:mrow>
</m:math>
</p>
</body>

</html>
```

Lors de la prévisualisation, ceci donnera:


$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$
MathPlayer
Version 2.0b
MathPlayer Home Page
MathPlayer™ is a product of Design Science, Inc.
Copyright ©2002-2004. All rights reserved.
The Team: Paul Topping, Robert Miner,
Peter Cooper, Doug Southam, William Kapes,
David Hart, Eduardo Tabacman,
Greg Langmead, Neil Soiffer
Design Science
How Science Communicates

20. SMIL

Le nom SMIL (Synchronized Multimedia Integration Language) est aussi issu du langage XML et permet de coordonner et synchroniser des flux multimédias (image, son, vidéo…) au sein d'un site web.

Comme un fichier HTML, un document SMIL contient deux parties distinctes: un en-tête et un corps, respectivement désignés par les éléments *header* et *footer*. Ces deux parteis devront définir les différentes composantes de l'animation.

1. La définition des régions contenant éléments multimédias
2. L'ordre et le timing d'apparition de ces objets. Il s'agit notamment de définir les déplacements et les transitions des éléments
3. La définition des éléments à utilisater en fonction des débits de connexion

Les exemples (essentiels) que nous allons faire sont tirés Cognitive Development Laboratory ainsi que les sources vidéos. Voici les fichiers nécessaires:

1. Black.rm (video montrant le joueur avec les pieces noires)
2. White.rm (video montrant le joueur avec les pieces blanches)
3. Board.rm (video montrant le plateau d'échecs)
4. Voicetrack.rm (fichier audio de la narration de jeu)

Il faut bien évidemment avoir un lecteur de fichier **.rm*. La version gratuite de *RealPlayer* (www.real.com) suffira pour nos exemples (WinAmp convient aussi mais pas MediaPlayer).

20.1.1 Timing

Nous allons dans cet exemple utiliser le fichier *board.rm* dont le temps réel est de 37 secondes et écourter le visionnement à l'aide du code suivant (dans XMLSpy vous pouvez aussi créer des fichiers smil).

```
<smil>
    <body>
        <video src="board.rm" clip-begin="16s" clip-end="27s"/>
    </body>

</smil>
```

que nous enregistrerons sous le nom *finalmove.smil*. En double cliquant sur ce fichier SMIL la vidéo sera jouée entre le temps 16s-27s.

20.1.2 MultipleCams

Nous allons cette fois jouer trois vidéos d'un seul coup. Enregistrez le code suivant dans un fichier smil que vous nommerez *threecams.smil*:

```
<smil>
    <head>
        <layout>
            <root-layout width="1080" height="240"/>
            <region id="video_left" width="360" height="240" left="0" top="0"/>
            <region id="video_center" width="360" height="240" left="360" top="0"/>
            <region id="video_right" width="360" height="240" left="720" top="0"/>
        </layout>
    </head>
    <body>
        <par dur="30s">
            <video src="black.rm" clip-begin="1.09s" region="video_left"/>
            <video src="board.rm" clip-begin="0s" region="video_center"/>
            <video src="white.rm" clip-begin="1.10s" region="video_right"/>
        </par>
    </body>

</smil>
```

En double cliquant sur ce fichier SMIL, les trois vidéos seront jouées en même temps dans des cadres et de temps défini par le fichier SMIL.

20.1.3 Ajout de sons

Dans cet exemple, nous allons jouer un fichier son au-dessus de la video. Enregistrez le code suivant dans un fichier smil que vous nommerez *voiceover.smil*:

```
<smil>
    <head>
        <layout>
            <root-layout width="1080" height="240"/>
            <region id="video_left" width="360" height="240" left="0" top="0"/>
            <region id="video_center" width="360" height="240" left="360" top="0"/>
            <region id="video_right" width="360" height="240" left="720" top="0"/>
```

```
        </layout>
    </head>
    <body>
        <par dur="35s">
            <video src="black.rm" begin="5s" clip-begin="1.09s" region="video_left"/>
            <video src="board.rm" begin="5s" clip-begin="0s" region="video_center"/>
            <video src="white.rm" begin="5s" clip-begin="1.10s" region="video_right"/>
            <audio src="voicetrack.rm"/>
        </par>
    </body>

</smil>
```

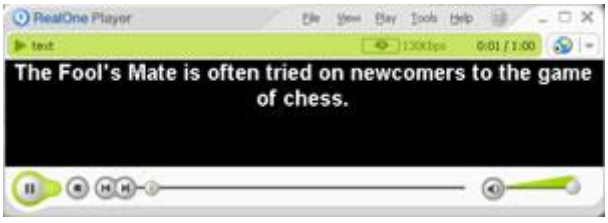
En double cliquant sur ce fichier SMIL, les trois vidéos seront jouées en même temps avec des commentaires supplémentaires.

20.1.4 Ajout de texte

Maintenant, créez un fichier *text.rt* avec le code suivant:

```
<window width="560"
      height="100"
      bgcolor="black"/>

<font color="white" face="arial" size="+2">
<b>
<center>

<time begin="00:00:00.00"/><clear/>
The Fool's Mate is often tried on newcomers to the game of chess.

<time begin="00:00:05.00"/><clear/>
White begins, and opens up his King to a fatal attack.

<time begin="00:00:12.50"/><clear/>
Black moves a pawn to give her Queen room to move out.

<time begin="00:00:19.50"/><clear/>
White moves the other pawn forward, leaving a clear line of attack on the King.

<time begin="00:00:31.46"/><clear/>
Black checkmates using her Queen.

<time begin="00:00:34.90"/><clear/>
It is rarely a good idea to move the pawns on f2, g2 and h2 so early in the game as the King normally castles on this
side. If the pawns have been moved, they can no longer offer him adequate protection.
```

En double cliquant sur ce fichier rt (RealTime) le texte défilera à l'écran selon les temps indiqués dans le fichier ci-dessus:



20.1.5 Mixage audio, vidéo et texte

Dans cet exemple, nous allons combiner: les videos, le texte, et l'audio. Pour cela, enregistrez le code suivant sous le nom *voiceovertext.smil*:

```
<smil>
    <head>
        <layout>
            <root-layout width="1080" height="350"/>
            <region id="video_left" width="360" height="240" left="0" top="0"/>
            <region id="video_center" width="360" height="240" left="360" top="0"/>
            <region id="video_right" width="360" height="240" left="720" top="0"/>
            <region id="text_subtitle" width="560" height="100" left="260" top="250"/>
        </layout>
    </head>
    <body>
        <par dur="55s">
            <video src="black.rm" begin="5s" clip-begin="1.09s" region="video_left"/>
            <video src="board.rm" begin="5s" clip-begin="0s" region="video_center"/>
            <video src="white.rm" begin="5s" clip-begin="1.10s" region="video_right"/>
            <textstream src="text.rt" region="text_subtitle"/>
            <audio src="voicetrack.rm"/>
        </par>
    </body>

</smil>
```

En double cliquant sur ce fichier SMIL, les trois vidéos seront jouées en même temps avec des commentaires supplémentaires et en plus un texte !:

20.1.6 Alternement

Nous allons maintenant alterner les différentes vidéos avec les commentaires audio. Enregistrez le code suivant un fichier nommé *edited.smil*:

```
<smil>
    <head>
        <layout>
            <root-layout width="360" height="240"/>
            <region id="video_main" width="360" height="240" left="0" top="0"/>
        </layout>
    </head>
    <body>
        <par>
            <seq>
                <video src="black.rm" begin="2.1s" clip-begin="0s" dur="2.1s" region="video_main"/>
                <video src="board.rm" clip-begin="0s" dur="4.2s" region="video_main"/>
                <video src="black.rm" clip-begin="5s" dur="3.2s" region="video_main"/>
                <video src="board.rm" clip-begin="8s" dur="4.5s" region="video_main"/>
                <video src="white.rm" clip-begin="13.5s" dur="2s" region="video_main"/>
                <video src="board.rm" clip-begin="14s" dur="3.7s" region="video_main"/>
                <video src="black.rm" clip-begin="18s" dur="6s" region="video_main"/>
                <video src="board.rm" clip-begin="23s" dur="3.5s" region="video_main"/>
                <video src="white.rm" clip-begin="27s" dur="4.5s" region="video_main"/>
                <video src="board.rm" clip-begin="29s" dur="0.5s" region="video_main" fill="freeze"/>
```

```
            </seq>
            <seq>
                <audio src="voicetrack.rm"/>
            </seq>
        </par>
    </body>
</smil>
```

En double cliquant sur ce fichier SMIL, les trois vidéos seront jouées alternativement avec des commentaires audio.

20.1.7 Ajout d'images

Nous allons voir maintenant comment ajouter une image au début de notre animation. Ecrivez le code suivant dans un fichier *title3cams.smil*:

```
<smil>
    <head>
        <layout>
            <root-layout width="1080" height="350"/>
            <region id="video_left" width="360" height="240" left="0" top="0"/>
            <region id="video_center" width="360" height="240" left="360" top="0"/>
            <region id="video_right" width="360" height="240" left="720" top="0"/>
            <region id="text_subtitle" width="560" height="100" left="260" top="250"/>
        </layout>
    </head>
    <body>
        <par dur="55s">
            <video src="black.rm" begin="5s" clip-begin="1.09s" region="video_left"/>
            <seq>
                <img src="title.gif" dur="5s" region="video_center"/>
                <video src="board.rm" clip-begin="0s" region="video_center"/>
            </seq>
            <video src="white.rm" begin="5s" clip-begin="1.10s" region="video_right"/>
            <textstream src="text.rt" region="text_subtitle"/>
            <audio src="voicetrack.rm"/>
        </par>
    </body>

</smil>
```

En double cliquant sur ce fichier SMIL, les trois vidéos seront jouées des commentaires audio, du texte et une image au centre.

20.1.8 Effets de transition

Nous allons voir comment faire des effets de transition dans cet exemple. Pour cela, écrivez le code SMIL suivant dans un fichiez que vous nommerez *transisitions.smil*:

```
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN" "http://www.w3.org/2001/SMIL20/SMIL20.dtd">
```

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
    <head>
        <layout>
            <root-layout width="360" height="240"/>
            <region id="video_region" width="360" height="240" left="0" top="0" fit="meet"/>
        </layout>
        <transition id="tin" type="fade" subtype="fadeFromColor" fadeColor="black" dur="2s"/>
        <transition id="tout" type="fade" subtype="fadeToColor" fadeColor="black" dur="2s"/>
    </head>
    <body>
        <seq>
            <video transIn="tin" src="board.rm" clipBegin="7s" dur="5s" region="video_region" fill="remove"
transOut="tout"/>
        </seq>
    </body>
</smil>
```

En double cliquant sur ce fichier SMIL, une vidéo sera jouée avec un effet d'entrée et un de sortie.

21. SVG

SVG (Standard Vectro Graphic) est un langage XML dédié au graphisme vectoriel. La définition de l'image se fait à l'aide d'une série d'instructions (équations des courbes, codes de couleur, coordonnées des figures), et non par définition des pixels (points d'écrans) qui la représentent. C'est certainement un des enfants de XML les plus compliqués pour des non-ingénieurs car faisant souvent appel à des maths.

Normalisé par le W3C sur la base de plusieurs essais, notamment le Vector Markup Language (VML) de Microsoft et le Portable Graphic Markup Language d'Adobe, le langage SVG fournit une solution légère pour la mise en place de graphismes complexes.

Les objets graphiques peuvent être regroupés, stylisés, transformés et composés dans des objets précédemment rendus. L'ensemble de fonctions comprend des transformations imbriquées, des tracés de rognage, des masques basés sur le couche alpha et des objets de gabarit.

Pour visualiser les images faites en SVG, au même type que MathML, vous devrez charger sur le site d'Adobe le plug-in suivant pour votre navigateur:

http://www.adobe.com/svg/viewer/install/

ou, pour le meilleur, installer Adobe Illustrator sur votre machine (mais MS Visio, AutoCAD, vont aussi)!

Si vous créez un nouveau document SVG dans XMLSpy, vous aurez:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-flat-20030114.dtd">
<svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%">
    <desc>
        <!-- put a description here -->
    </desc>
    <g>
        <!-- your graphic here -->
    </g>

</svg>
```

Si vous souhaitez inclure une image SVG (*test.svg*) dans un fichier HTML voici comment il faudra procéder:

```
<html>
<body>
  <embed src="test.svg" width="500" height="500" type="image/svg+xml" />
</body>
</html>
```

Les exemples donnés ici sont pris du site de référence http://www.w3schools.com

21.1.1 Rectangles

Voici le code d'un premier exemple (à enregistrer sous le nom *Exemple1.svg*):

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
```

```
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<!--ici nous définissons la taille de zone de traçage-->
<svg version="1.1" xmlns="http://www.w3.org/2000/svg">
<!-- ici le rectange avec bordure que nous souhaitons dessiner-->
<g>
    <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:1;stroke:rgb(0,0,0)"/>
</g>
</svg>
```

qui donnera dans Internet Explorer (remarquez la zone blanche en différence avec le fond gris par défaut de IE):

à vous ensuite d'insérer (comme test) ce fichier SVG dans un fichier HTML selon la syntaxe décrite plus haut.

Voici un autre exemple un peu plus évolué:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1" xmlns="http://www.w3.org/2000/svg">
<g>
    <rect x="20" y="20" width="250" height="250" style="fill:blue;stroke:pink;stroke-width:5;fill-opacity:0.1;stroke-
opacity:0.9"/>
</g>
</svg>
```

qui donnera dans IE:

et cette fois avec des coins arrondis:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1" xmlns="http://www.w3.org/2000/svg">
<g>
    <rect x="20" y="20" rx="20" ry="20" width="250" height="100" style="fill:red;stroke:black;stroke-width:5;opacity:0.5"/>
</g>
</svg>
```

qui donnera dans IE:

ou encore mieux:

```
<svg width="500" height="150">
<g>
    <linearGradient id="backgroundGradient" x1="0" y1="0" x2="0" y2="120" gradientUnits="userSpaceOnUse">
        <stop offset="0%" style="stop-color:black"/>
        <stop offset="50%" style="stop-color:white"/>
        <stop offset="100%" style="stop-color:black"/>
    </linearGradient>
    <rect width="240" height="120" style="fill:url(#backgroundGradient);"/>
    <text x="80" y="140" style="fill:black;font-family:Verdana;font-size:9;text-anchor:middle;"
transform="translate(120,0)">
        <tspan>Exemple de graphisme SVG.</tspan>
    </text>
</g>
</svg>
```

Ce qui donnera dans IE:

Exemple de graphisme SVG.

Remarque: les coordonnées x,y d'un texte sont données par rapport au centre de l'image svg widht et height.

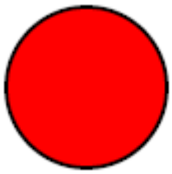
21.1.2 Cercle

Nous avons vu donc comment créer quelques rectangles simples et un peu plus stylés. Nous allons voir maintenant comment faire des cercles (au besoin, vous pouvez vous amuser a faire des cercles + des rectangles dans le même SVB bien sûr !).

Voici d'abord ce qu'il est possible de faire de plus simple:

```
<?xml version="1.0" standalone="no"?>
<svg width="300" height="300" version="1.1" xmlns="http://www.w3.org/2000/svg">
<g>
    <circle cx="100" cy="50" r="40" stroke="black" stroke-width="2" fill="red"/>
</g>
</svg>
```

ce qui donne:



Remarque: à vous de définir plusieurs cercles dans la même image si vous le désirez (selon vos paramètres ils se superposeront)

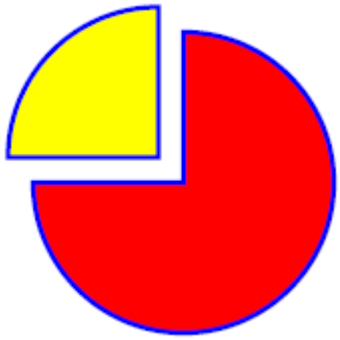
21.1.3 Arcs de cerlces

Il s'agit d'un exemple légerement plus compliqué… la création d'un camembert.

Le code SVG est le suivant:

```
<?xml version="1.0" standalone="no"?>
<svg width="12cm" height="5.25cm" viewBox="0 0 1200 400"
    xmlns="http://www.w3.org/2000/svg">
  <desc>Image d'un diagramme en "camembert" en deux parts</desc>
  <g>
  <path d="M300,200 h-150 a150,150 0 1,0 150,-150 z" fill="red" stroke="blue" stroke-width="5" />
  <path d="M275,175 v-150 a150,150 0 0,0 -150,150 z" fill="yellow" stroke="blue" stroke-width="5" />
  </g>
</svg>
```

et donne le résultat suivant:



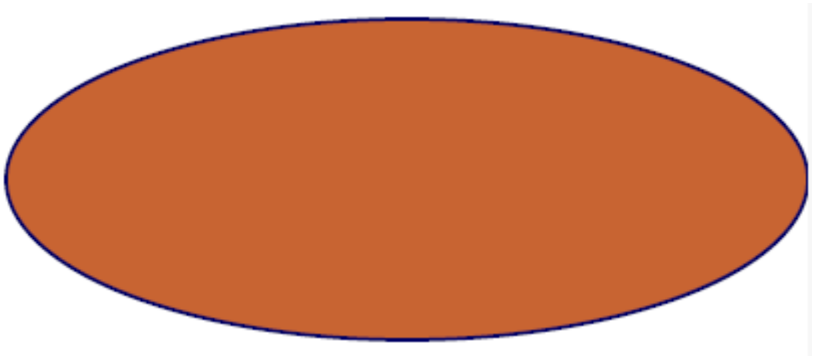
Pour voir quels sont les rôles des différentes valeurs du fromage, il suffit de jouer avec!

21.1.4 Ellipse

Dans la suite logique, passons aux ellipses. Voici encore un exemple simple:

```
<?xml version="1.0" standalone="no"?>
<svg width="500" height="500" version="1.1" xmlns="http://www.w3.org/2000/svg">
<g>
   <ellipse cx="300" cy="150" rx="200" ry="80" style="fill:rgb(200,100,50); stroke:rgb(0,0,100);stroke-width:2"/>
</g>
</svg>
```

Ce qui donne:



Remarque: à vous de définir plusieurs ellipses dans la même image si vous le désirez (selon vos paramètres elles se superposeront).

21.1.5 Ligne

La ligne est certainement l'un des éléments les plus importants en SVG pour la D.A.O ou le dessins de contours d'objets par segements (cartes, etc.). Le code de création est simple:

```
<?xml version="1.0" standalone="no"?>
<svg width="300" height="300" version="1.1" xmlns="http://www.w3.org/2000/svg">
<g>
    <line x1="0" y1="0" x2="300" y2="300" style="stroke:rgb(99,99,99);stroke-width:2"/>
</g>
</svg>
```

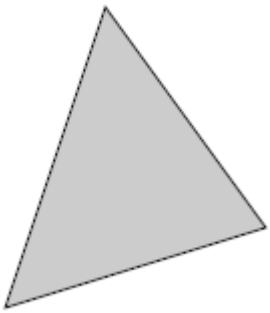
Ce qui donne:

21.1.6 Polygones

Un polygone est par définition une suite d'au moins trois points définissant des segments croisés ou non donnant une surface fermée. Par exemple:

```
<?xml version="1.0" standalone="no"?>
<svg width="100%" height="100%" version="1.1" xmlns="http://www.w3.org/2000/svg">
<g>
    <polygon points="220,100 300,210 170,250" style="fill:#cccccc; stroke:#000000;stroke-width:1"/>
</g>
</svg>
```

Ce qui donne:



21.1.7 Polylignes

Les polylignes ont une importance tout aussi grande que les polygones. Leur utilisation est très simple. Voyons un exemple:

```
<?xml version="1.0" standalone="no"?>
<svg width="100%" height="100%" version="1.1" xmlns="http://www.w3.org/2000/svg">
<g>
    <polyline points="0,0 0,20 20,20 20,40 40,40 40,60" style="fill:white;stroke:red;stroke-width:2"/>
</g>
</svg>
```

Ce qui donne:

21.1.8 XSL/XML et SVG

Nous allons dans l'exemple suivant montrer comment intégrer du SVG avec XML et XSL. Le lecteur pourra en tant qu'exercice complexifier l'exemple pour créer plusieurs cercles automatiquement en fonction du contenu de son fichier XML.

Considérons le fichier XML suivant:

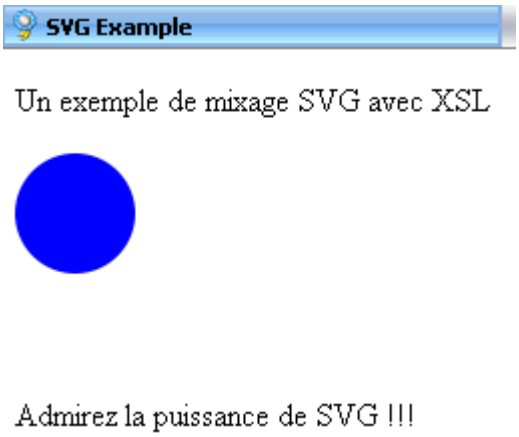
```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="MiseEnForme.xsl"?>
<root>
    <rayon>30</rayon>
</root>
```

Nous allons utiliser le nœud *rayon* pour dessiner un cercle à l'aide de SVG et XSL.

Voici le fichier XSL dont la première partie un peu technique n'est en fait que l'inclusion du plug-in Adobe SVG dans le fichier XHTML rendu!

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html xmlns:svg="http://www.w3.org/2000/svg">
    <!-- on déclare le plug-in SVG-->
    <object id="AdobeSVG"
      CLASSID="clsid:78156a80-c6a1-4bbf-8e6a-3cd390eeb4e2">
    </object>
    <xsl:processing-instruction name = "import" >
       namespace="svg" implementation="#AdobeSVG"
    </xsl:processing-instruction>
    <!-- déclaration terminée-->
    <head><title>SVG Example</title></head>
    <body>
    <p>
      Un exemple de mixage SVG avec XSL
    </p>
    <xsl:for-each select="root">
        <svg:svg width="100px" height="100px" viewBox="0 0 100 100">
        <svg:circle cx="30" cy="30" r="{rayon}" fill="blue" stroke="none"/>
        </svg:svg>
    </xsl:for-each>
    <p>
        Admirez la puissance de SVG !!!
    </p>
    </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Le résultat obtenu sera alors:


SVG Example
Un exemple de mixage SVG avec XSL
Admirez la puissance de SVG !!!

22. XPath

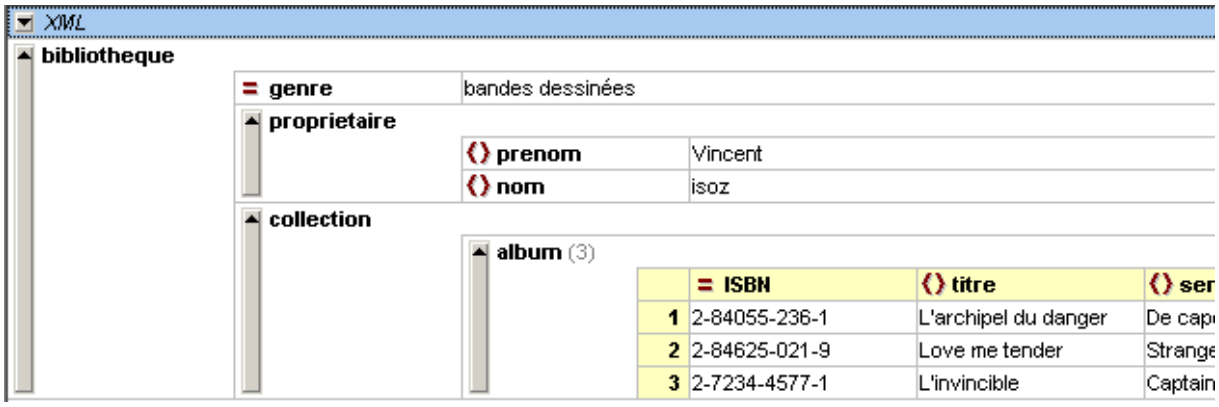
XPath (pour XML Language 1.0) est un lange abstrait (on ne peut l'utiliser qu'en employant la syntaxe d'un autre langage, comme XSLT, JavaScript ou C#...) défini par le W3C pour l'adressage et et la sélection d'objets XML au sein d'un document. Nous entendons par "adressage" la capacité d'établir un lien logique au sein d'un document entre un élément et au autre à l'aide d'une expression appelée "adresse". Sur la base du modèle arborescent d'un document, les expressions XPath sont évaluées par le parseur XML et permettent d'identifier certains nœuds XML. Ces nœuds dit "sélectionnés" sont regroupés dans un ensemble de nœuds appelé "node-set"

Pour voir quelques commandes XPath, nous allons utiliser le code XML suivant à enregistrer sous nom *bibliotheque.xml*:

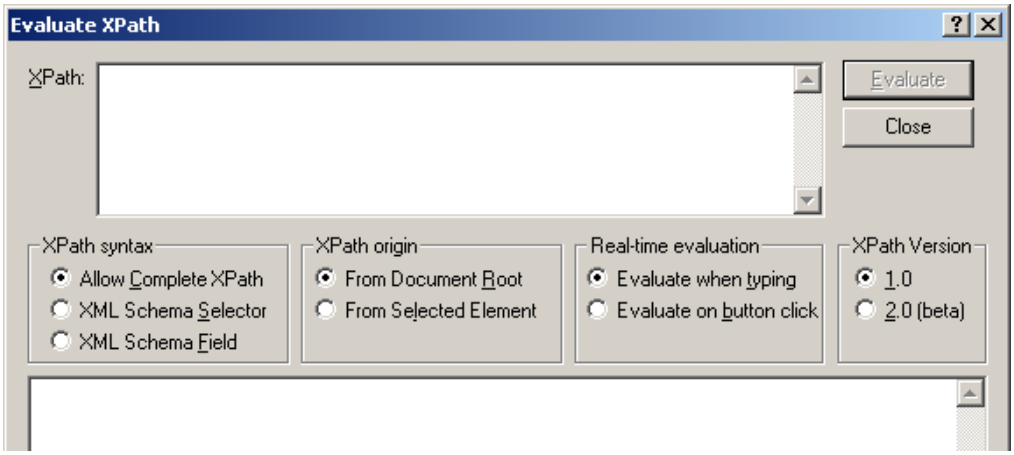
```
<?xml version="1.0" encoding="iso-8859-1"?>

<bibliotheque genre="bandes dessinées">
    <proprietaire>
        <prenom>Vincent</prenom>
        <nom>isoz</nom>
    </proprietaire>
    <collection>
        <album ISBN="2-84055-236-1">
            <titre>L'archipel du danger</titre>
            <serie>De cape et de crocs</serie>
            <numero>3</numero>
            <auteur role="scenariste">
                <prenom>Alain</prenom>
                <nom>Ayroles</nom>
            </auteur>
            <auteur role="dessinateur">
                <prenom>Jean-Luc</prenom>
                <nom>Masbou</nom>
            </auteur>
            <editeur>Delcourt</editeur>
            <prix>11</prix>
            <cote>11</cote>
        </album>
        <album ISBN="2-84625-021-9">
            <titre>Love me tender</titre>
            <serie>Strangers in paradise</serie>
            <numero>4</numero>
            <auteur role="tous">
                <prenom>Terry</prenom>
                <nom>Moore</nom>
            </auteur>
            <editeur>Bulle Dog</editeur>
            <prix>8</prix>
            <cote>9</cote>
        </album>
        <album ISBN="2-7234-4577-1">
            <titre>L'invincible</titre>
            <serie>Captain Biceps</serie>
            <numero>1</numero>
            <auteur role="scenariste">
                <pseudo>Zep</pseudo>
            </auteur>
            <editeur>Glénat</editeur>
            <prix>12</prix>
            <cote>14</cote>
        </album>
    </collection>

</bibliotheque>
```

Nous allons voir à quoi ressemble ce langage XPath en dehors de tout langage de programmation. Pour cela, nous allons à nouveau utiliser XMLSpy.

La représentation du fichier XML ci-dessous sous la forme arborescente (apelée "infoset") est la suivante (vous devez activer la vue *Grid* dans XMLSpy):



Ensuite, cliquez sur le bouton  . La fenêtre suivante apparaît:



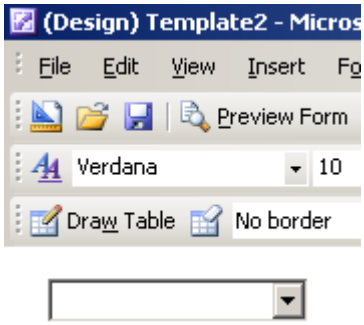
L'idée est de saisir quelque chose dans le champ *XPath* et de cliquer sur le bouton *Evaluate*. Voici quelques requêtes que vous pouvez saisir:

01.	/
02.	bibliotheque
03.	bibliotheque/collection
04.	bibliotheque/collection/*
05.	bibliotheque/collection/album
06.	//album
07.	//prenom
08.	//auteur/@role
09.	//@*
10.	//auteur/nom
11.	//nom
12.	//nom/..
13.	bibliotheque/collection/album[1]

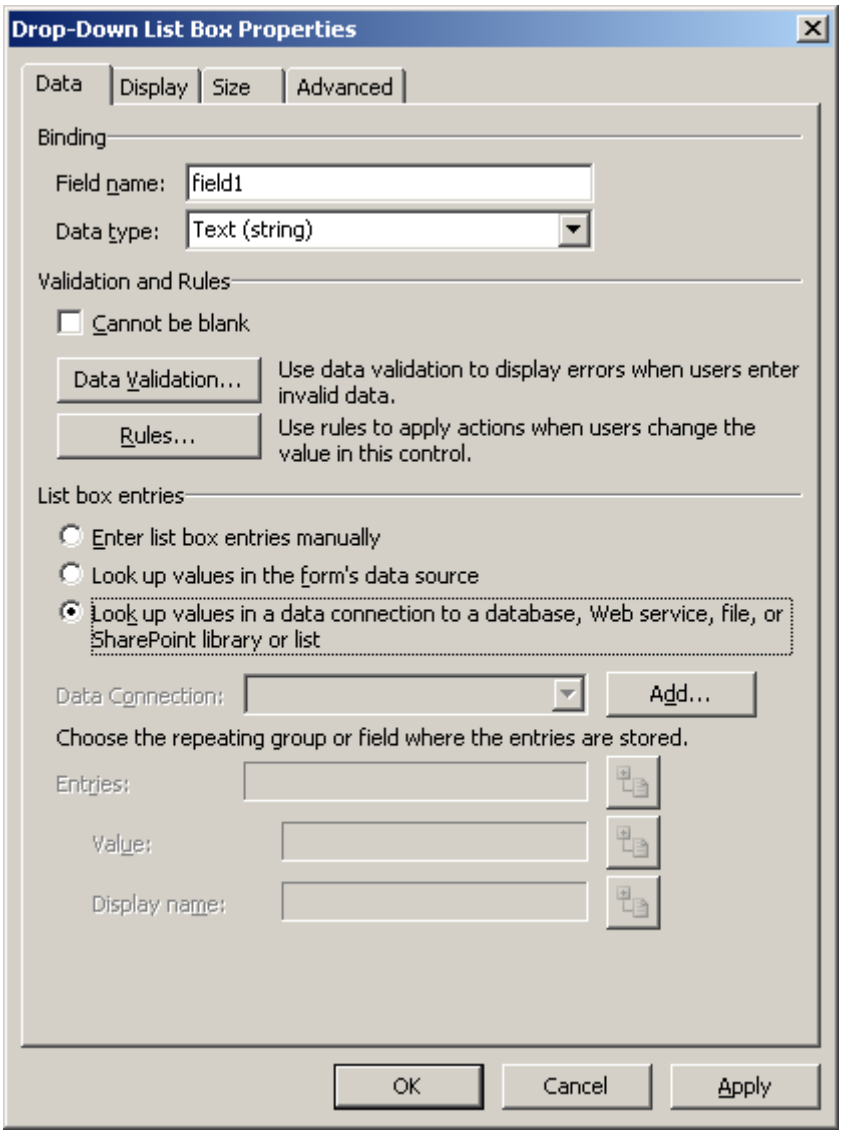
14.	//album[1]
15.	//album[last()]
16.	//albume/auteur[nom!='Masbou']/nom
17.	//album[3]/auteur[1]/pseudo
18.	//album[@ISBN="2-84055-236-1"]
19.	//auteur[nom]
20.	//album[prix * "0.196" < "2"]
21.	//album[prix < cote]
22.	//auteur[@role="dessinateur" or @role="tous"]
23.	contains(//album[1]/auteur[1]/@role,"scenariste")

Ce langage est vaste mais nous donnons ici les principaux elements.

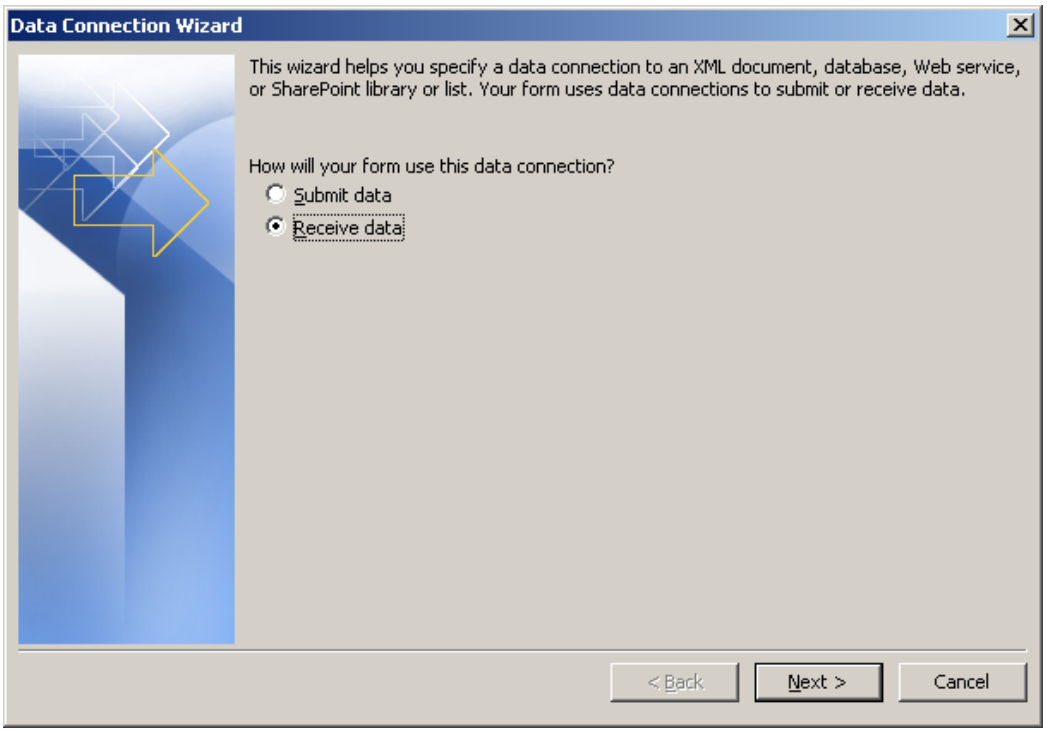
XPath est aussi beaucoup utilisé dans InfoPath, pour exemple, créons un nouveau formulaire vide et ajoutons-y une liste déroulante:



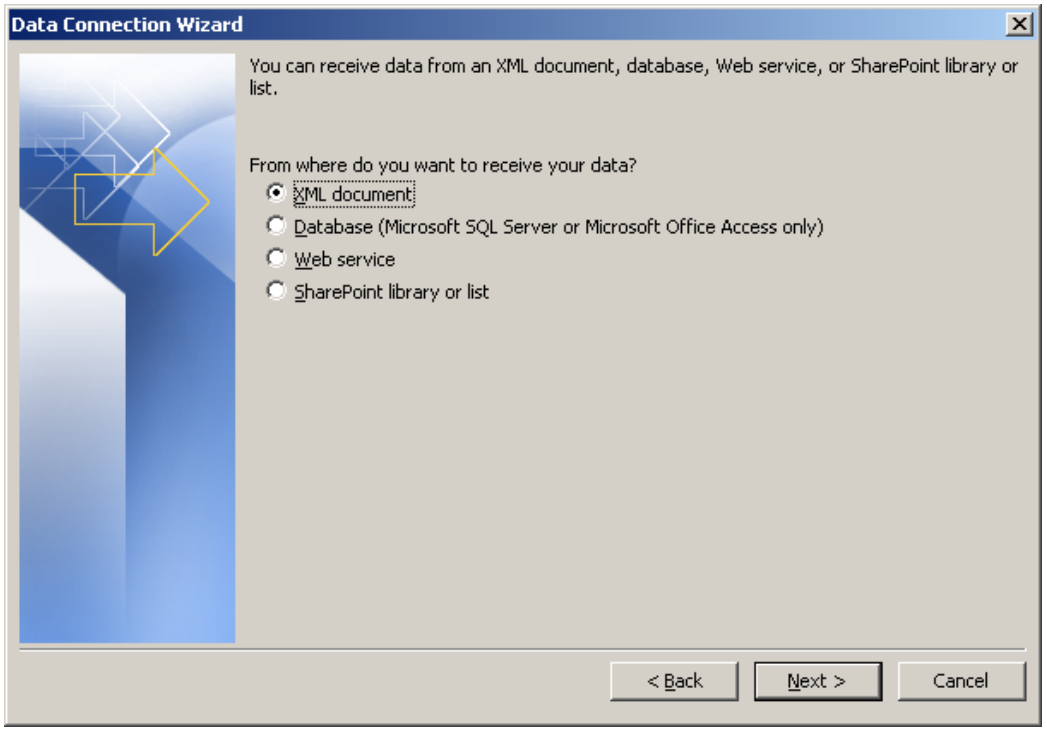
Faites double clic sur la list déroulante choisissez l'option *Look up values in a data*.



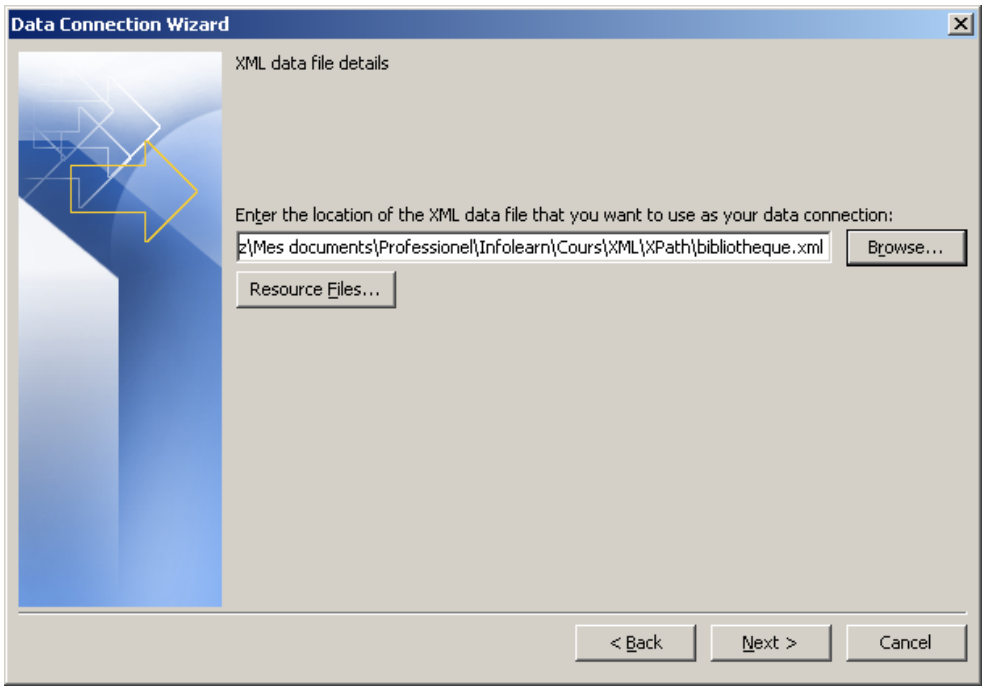
cliquez sur *Add* et ensuite:



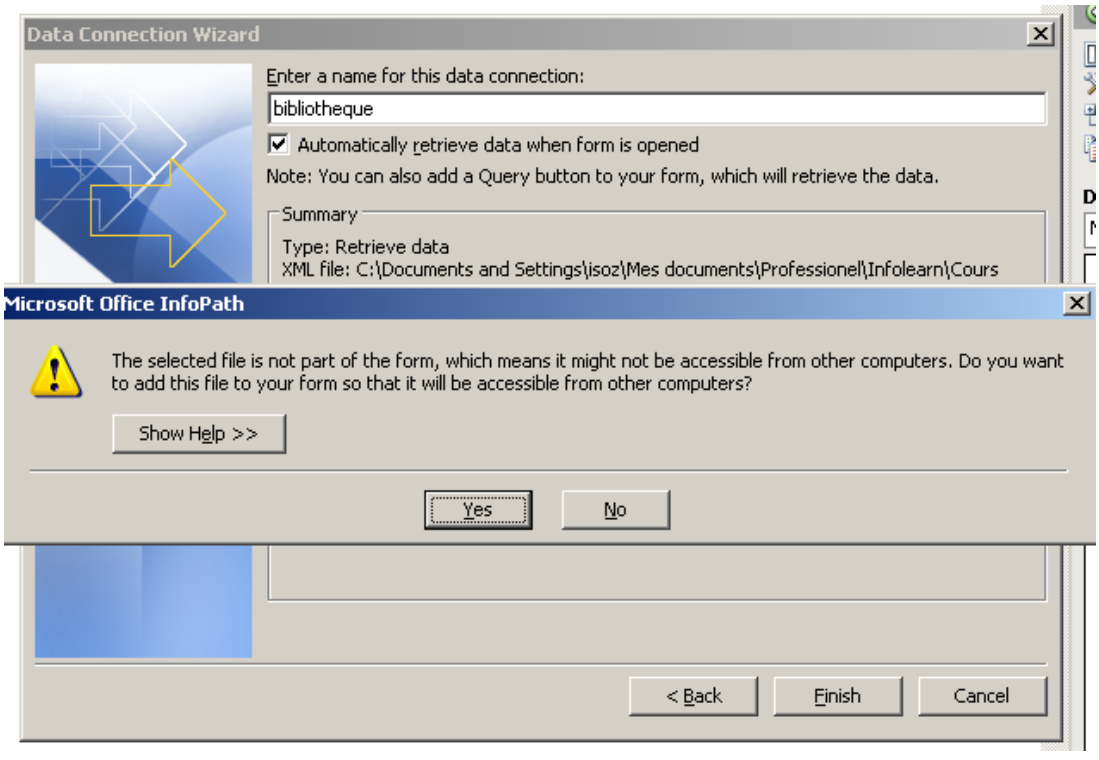
Cliquez sur *Next*:



Encore sur *Next* et allez chercher notre fichier de *bibliotheque.xml*:

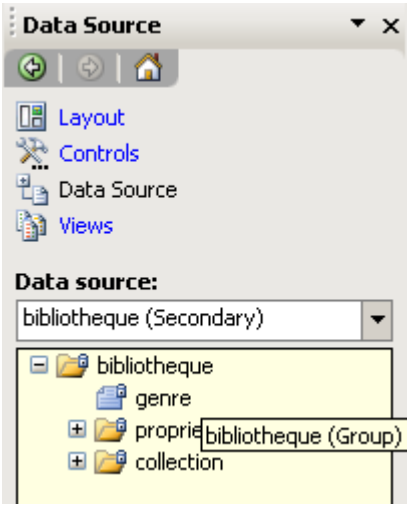


Cliquez sur *Next* et *Finish*. Infopath vous demande si vous souhaitez intégrer[3] le fichier *XML* au formulaire. Dites *Yes*:

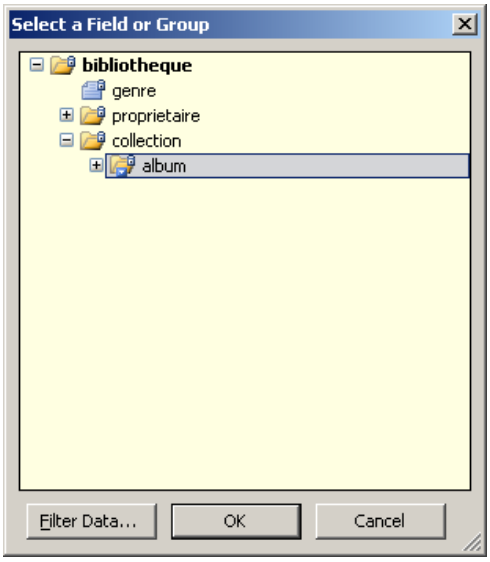


Ensuite, dans le volet MS Office allez dans *Data Source* et sélectionnez la source de données créée précédemment:

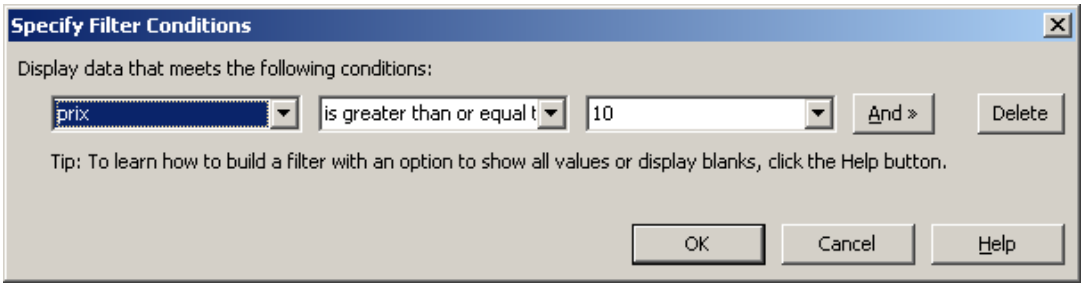
[3] On peut alors y accéder par le menu *Tools/Resource Files*



Ensuite, cliquez sur  au niveau des *Entries* et choisissez:



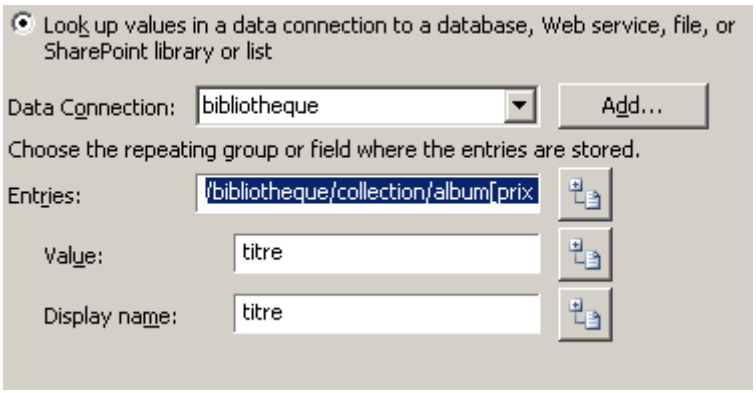
et dans *Filter Data* créez le fitlre:



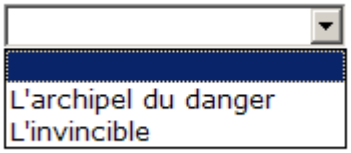
Si vous validez par *OK*, vous aurez comme requête XPath:

/bibliotheque/collection/album[prix >= 10]

Ensuite libre à vous d'y mettre:



Ce qui nous renverra bien deux albums:



22.1 Exercices: Football

Considérons le fichier xml suivant (votre formateur vous mettra la version complète à disposition):

```
<?xml version="1.0" standalone="no"?>
<CHAMPIONNAT DIVISION="1" SAISON="2003-2004">
    <JOURNEE NUMERO="1" DATE="01/08/2003">
        <RENCONTRE DOMICILE="Auxerre" EXTERIEUR="Nice" SCORED="1" SCOREE="2"/>
        <RENCONTRE DOMICILE="Guingamp" EXTERIEUR="Marseille" SCORED="0" SCOREE="1"/>
        <RENCONTRE DOMICILE="Lens" EXTERIEUR="LeMans" SCORED="0" SCOREE="0"/>
        <RENCONTRE DOMICILE="Lille" EXTERIEUR="Lyon" SCORED="1" SCOREE="0"/>
        <RENCONTRE DOMICILE="Metz" EXTERIEUR="Ajaccio" SCORED="0" SCOREE="1"/>
        <RENCONTRE DOMICILE="Monaco" EXTERIEUR="Bordeaux" SCORED="2" SCOREE="0"/>
        <RENCONTRE DOMICILE="Montpellier" EXTERIEUR="Rennes" SCORED="1" SCOREE="1"/>
        <RENCONTRE DOMICILE="ParisSG" EXTERIEUR="Bastia" SCORED="0" SCOREE="0"/>
        <RENCONTRE DOMICILE="Sochaux" EXTERIEUR="Nantes" SCORED="2" SCOREE="1"/>
        <RENCONTRE DOMICILE="Toulouse" EXTERIEUR="Strasbourg" SCORED="1" SCOREE="1"/>
    </JOURNEE>
    <JOURNEE NUMERO="2" DATE="08/08/2003">
        <RENCONTRE DOMICILE="Bastia" EXTERIEUR="Metz" SCORED="0" SCOREE="2"/>
        <RENCONTRE DOMICILE="Bordeaux" EXTERIEUR="Montpellier" SCORED="0" SCOREE="1"/>
        <RENCONTRE DOMICILE="LeMans" EXTERIEUR="Ajaccio" SCORED="0" SCOREE="1"/>
        <RENCONTRE DOMICILE="Lille" EXTERIEUR="ParisSG" SCORED="1" SCOREE="0"/>
        <RENCONTRE DOMICILE="Lyon" EXTERIEUR="Monaco" SCORED="3" SCOREE="1"/>
        <RENCONTRE DOMICILE="Marseille" EXTERIEUR="Auxerre" SCORED="1" SCOREE="0"/>
        <RENCONTRE DOMICILE="Nantes" EXTERIEUR="Lens" SCORED="2" SCOREE="0"/>
        <RENCONTRE DOMICILE="Nice" EXTERIEUR="Sochaux" SCORED="1" SCOREE="0"/>
        <RENCONTRE DOMICILE="Rennes" EXTERIEUR="Toulouse" SCORED="1" SCOREE="0"/>
        <RENCONTRE DOMICILE="Strasbourg" EXTERIEUR="Guingamp" SCORED="2" SCOREE="0"/>
    </JOURNEE>
    <JOURNEE NUMERO="3" DATE="16/08/2003">
        <RENCONTRE DOMICILE="Ajaccio" EXTERIEUR="Nantes" SCORED="1" SCOREE="3"/>
        <RENCONTRE DOMICILE="Auxerre" EXTERIEUR="Strasbourg" SCORED="3" SCOREE="2"/>
        <RENCONTRE DOMICILE="Guingamp" EXTERIEUR="Bordeaux" SCORED="1" SCOREE="3"/>
        <RENCONTRE DOMICILE="LeMans" EXTERIEUR="Nice" SCORED="1" SCOREE="1"/>
        <RENCONTRE DOMICILE="Lens" EXTERIEUR="Marseille" SCORED="2" SCOREE="1"/>
        <RENCONTRE DOMICILE="Metz" EXTERIEUR="ParisSG" SCORED="0" SCOREE="1"/>
        <RENCONTRE DOMICILE="Monaco" EXTERIEUR="Bastia" SCORED="2" SCOREE="0"/>
        <RENCONTRE DOMICILE="Montpellier" EXTERIEUR="Lyon" SCORED="0" SCOREE="2"/>
        <RENCONTRE DOMICILE="Sochaux" EXTERIEUR="Rennes" SCORED="1" SCOREE="1"/>
        <RENCONTRE DOMICILE="Toulouse" EXTERIEUR="Lille" SCORED="0" SCOREE="3"/>
    </JOURNEE>
    <JOURNEE NUMERO="4" DATE="23/08/2003">
        <RENCONTRE DOMICILE="Bastia" EXTERIEUR="Montpellier" SCORED="1" SCOREE="0"/>
        <RENCONTRE DOMICILE="Bordeaux" EXTERIEUR="Auxerre" SCORED="2" SCOREE="0"/>
        <RENCONTRE DOMICILE="Lille" EXTERIEUR="Metz" SCORED="1" SCOREE="1"/>
        <RENCONTRE DOMICILE="Lyon" EXTERIEUR="Toulouse" SCORED="0" SCOREE="0"/>
        <RENCONTRE DOMICILE="Marseille" EXTERIEUR="Sochaux" SCORED="2" SCOREE="0"/>
        <RENCONTRE DOMICILE="Nantes" EXTERIEUR="LeMans" SCORED="1" SCOREE="0"/>
```

Nous vous demons à l'aide de XMLSpy et l'outil de modélisation de requêtes XPath de sortir les informations suivantes:

1. Les journées précédant la 8ème
2. Les rencontres de la 4ème journée
3. La première rencontre de chacune des journées
4. L'adversaire de Bastia lors de la journée 10

5. Tous les adversaires de Bastia
6. La liste des matches non joués
7. Les matches nuls de Bastia à domicile
8. Le nombre et le pourcentage de matches nuls sur l'ensemble du championnat

Voici les solutions (pour chaque question plusieurs méthodes sont possibles):

1. Les journées précédant la 8ème

```
/descendant::JOURNEE[attribute::NUMERO < 8]
//JOURNEE[@NUMERO < 8]

/descendant::JOURNEE[attribute::NUMERO=8]/preceding-sibling::JOURNEE
//JOURNEE[@NUMERO=8]/preceding-sibling::JOURNEE
```

2. Les rencontres de la 4ème journée

```
/descendant::JOURNEE[attribute::NUMERO=4]/child::RENCONTRE
//JOURNEE[@NUMERO=4]/RENCONTRE
```

3. La première rencontre de chacune des journées

```
/descendant::JOURNEE/child::RENCONTRE[position()=1]
//JOURNEE/RENCONTRE[1]
```

4. L'adversaire de Bastia lors de la journée 10

```
/descendant::JOURNEE[attribute::NUMERO=10]/child::RENCONTRE[attribute::DOMICILE='Bastia']/attribute::EXTERIEUR
/descendant::JOURNEE[attribute::NUMERO=10]/child::RENCONTRE[attribute::EXTERIEUR='Bastia']/attribute::DOMICILE
//JOURNEE[@NUMERO=10]/RENCONTRE[@DOMICILE='Bastia']/@EXTERIEUR
//JOURNEE[@NUMERO=10]/RENCONTRE[@EXTERIEUR='Bastia']/@DOMICILE
```

5. Tous les adversaires de Bastia

```
/descendant::RENCONTRE[attribute::DOMICILE='Bastia']/attribute::EXTERIEUR
/descendant::RENCONTRE[attribute::EXTERIEUR='Bastia']/attribute::DOMICILE
//RENCONTRE[@DOMICILE='Bastia']/@EXTERIEUR
//RENCONTRE[@EXTERIEUR='Bastia']/@DOMICILE
```

6. La liste des matches non joués

```
/descendant::RENCONTRE[attribute::SCORED='-']
//RENCONTRE[@SCORED='-']
```

7. Les matches nuls de Bastia à domicile

```
/descendant::RENCONTRE[attribute::DOMICILE='Bastia' and attribute::SCORED!='-' and attribute::SCORED=attribute::SCOREE]
//RENCONTRE[@DOMICILE='Bastia' and @SCORED!='-' and @SCORED=@SCOREE]
```

8. Le nombre et le pourcentage de matches nuls sur l'ensemble du championnat

```
count(/descendant::RENCONTRE[attribute::SCORED!='-' and attribute::SCORED=attribute::SCOREE])
count(//RENCONTRE[@SCORED!='-' and @SCORED=@SCOREE])

100.*count(/descendant::RENCONTRE[attribute::SCORED!='-' and attribute::SCORED=attribute::SCOREE]) div count(/descendant::RENCONTRE[attribute::SCORED!='-'])
100*count(//RENCONTRE[@SCORED!='-' and @SCORED=@SCOREE]) div count(//RENCONTRE[@SCORED!='-'])
```