

XML : UNE PONCTUATION A L'USAGE DES ORDINATEURS

Eric van der Vlist (vdv@dyomedea.com)

**XML : une ponctuation a l'usage des
ordinateurs**

Journée XML et STATISTIQUE

Juin 2004

Copies : <http://dyomedea.com/papers/2004-infostat/>

POURQUOI XML?

Donner aux programmes l'accès aux documents

L'objectif principal de XML et des tout langage de balisage est de donner aux programmes informatiques l'accès aux informations contenues dans les documents.

Prenons par exemple l' accusé de réception de commande suivant :

Eric Van-Der-Vlist, bonjour !

Nous avons le plaisir de vous confirmer la validation du paiement de votre commande numéro 2767895 du 03/11/2003.

Produits de votre commande :

Ces produits vous seront expédiés le 18/11/2003

Produit	:	Carte Wireless pour PC portables - XG511 en 54 Mbits
Référence	:	XG511SD
Quantité	:	1
Marque	:	NETSPEED

Produit	:	Disque dur Landwindow Squale 80 Go - 7200 Tr/min
Référence	:	sq80
Quantité	:	2
Marque	:	LANDWINDOW

Si vous voulez suivre l'évolution de votre commande ou y apporter des modifications, connectez-vous 24h/24 et 7j/7 à l'adresse suivante :

<http://www.ruedumarche.com/client/commande/>

Cette adresse vous permet d'avoir accès en temps réel à toutes les informations de votre commande : réception des règlements, délais estimés d'expédition, numero de colis chez nos transporteurs, changement éventuel de disponibilité ...

Vous pourrez également effectuer les opérations habituelles : demandes de duplicata de factures, modification de commande sur cet espace réservé.

Cordialement

L'équipe Service Client.

Bien que toutes les informations nécessaires soient présentes dans ce document et que tout lecteur humain (et francophone) le comprenne sans difficulté même s'il n'a lu aucun document ayant ce format, il est difficile à un programme informatique de comprendre qu'il s'agit d'un accusé de réception de commande et d'isoler les différentes informations qui le composent (nom de l'acheteur, date de commande, date d'expédition prévue, liste des produits, ...).

Par une ponctuation adaptée

La solution adoptée par XML est de rajouter une ponctuation spécifique, généralement appelée « balisage » (ou « markup » en anglais) qui identifie les informations présentes dans un document comme on le ferait d'un coup de stabilo sur un document papier :

```
<?xml version="1.0" encoding="utf-8"?>
<commande type="accusé de réception">
<client>Eric Van-Der-Vlist</client>, bonjour !
```

Nous avons le plaisir de vous confirmer la validation du paiement de votre commande numéro <numéro>2767895</numéro> du <date-commande>03/11/2003</date-commande>.

```
<articles>
Produits de votre commande :
-----
Ces produits vous seront expédiés le <date-expédition
type="prévue">18/11/2003</date-expédition>
<article>
  <produit>Carte Wireless pour PC portables - XG511 en 54
Mbits</produit>
  <référence>XG511SD</référence>
  <quantité>1</quantité>
  <marque>NETSPEED</marque>
</article>
```

```
<article>
  <produit>Disque dur Landwindow Squale 80 Go - 7200 Tr/min</produit>
  <référence>sq80</référence>
  <quantité>2</quantité>
  <marque>LANDWINDOW</marque>
</article>
</articles>
```

Si vous voulez suivre l'évolution de votre commande ou y apporter des modifications, connectez-vous 24h/24 et 7j/7 à l'adresse suivante :

```
<suivi href="http://www.ruedumarche.com/client/commande/">
```

Cette adresse vous permet d'avoir accès en temps réel à toutes les informations de votre commande : réception des règlements, délais estimés d'expédition, numero de colis chez nos transporteurs, changement éventuel de disponibilité ...

Vous pourrez également effectuer les opérations habituelles : demandes de duplicata de factures, modification de commande sur cet espace réservé.
</suivi>

Cordialement

<émetteur>L'équipe Service Client.</émetteur>
</commande>

Suite d'une longue évolution

Cette ponctuation peut être vue comme le dernier épisode d'un processus commencé lorsque les romains ont introduit la ponctuation dans les textes. Notre commande rédigée en grec ancien (avec nos caractères actuels) aurait en effet donné quelque chose qui aurait ressemblé à :

```
ERICVANDERVLISTBONJOURNOUSAVONSLEPLAISIRDEVOUSCONFIRMERLAVALIDATIOND  
UPAIEMENTDEVOTRECOMMANDENUMÉRO2767895DU03112003PRODUITSDEVOTRECOMMAN  
DECESPRODUITSVOUSSEONTEXPÉDIÉSLE18112003PRODUITCARTEWIRELESSPOURPCP  
ORTABLESXG511EN54MBITSRÉFÉRENCEXG511SDQUANTITÉ1MARQUENETSPEEDPRODUIT  
DISQUEDURLANDWINDOWSQUALE80GO7200TRMINRÉFÉRENCESQ80QUANTITÉ2MARQUELA  
NDWINDOWSIVOUSVOULEZSUIVRELÉVOLUTIONDEVOTRECOMMANDEOUYAPPORTERDESMOD  
IFICATIONSCONNECTEZVOUS24H24ET7J7ÀLADRESSESUIVANTEHTTPWWWRUEDUMARCHE  
COMCLIENCOMMANDECETTEADRESSEVOUSPERMETDAVOIRACCÈSENTEMPSRÉELÀTOUTES  
LESINFORMATIONSDEVOTRECOMMANDERECEPTIONDESREGLLEMENTSDÉLAISESTIMÉSDEX  
PÉDITIONNUMERODECOLISCHEZNOSTRANSPORTEURSCHANGEMENTÉVENTUELDEDISPONI  
BILITÉVOUSPOURRÉGALEMENTEFFECTUERLESOPÉRATIONSHABITUELLESDEMANDES  
EDUPLICATADEFACTURESMODIFICATIONDECOMMANDESURCETESPACERÉSERVÉCORDIAL  
EMENTLÉQUIPESERVICECLIENT
```

XML : DÉFINITIONS

Constituants d'un document XML

La structure d'un document XML est marquée par ses balises (« tag » en anglais).

Balise de début

Les balises de début commencent par « < » et se terminent par « > », par exemple :

```
<date-expédition type="prévue">
```

Balise de fin

Les balises de fin commencent par « </ » et se terminent par « > », par exemple :

```
</quantité>
```

Balise vide

Les balises de début commencent par « < » et se terminent par « /> », par exemple :

```
<br/>
```

Élément

Un élément est l'ensemble des informations comprises entre une balise de début et la balise de fin correspondante, par exemple :

```
<référence>XG511SD</référence>
```

Lorsqu'il n'y a pas d'information entre la balise de début et la balise de fin, l'élément est dit « vide » et il peut être représenté par une balise vide :

```
<br/>
```

Attribut

Les attributs sont des métadonnées placées dans les balises de début, par exemple :

```
type="prévue"
```

Styles de XML

XML orienté document

Notre exemple de commande mélange texte et éléments :

```
<articles>
Produits de votre commande :
-----
Ces produits vous seront expédiés le <date-expédition
type="prévue">18/11/2003</date-expédition>
```

Ce style de structure XML est qualifiée de « XML orienté document ».

XML orienté données

Si l'on réduit cet exemple pour que le contenu de chaque élément comprenne soit du texte soit des sous-éléments :

```
<articles>
  <date-expédition type="prévue">18/11/2003</date-expédition>
```

on obtient une structure XML qualifiée de « XML orienté données ».

SGML SUR LE WEB

XML est un sous ensemble de SGML

Entre l'introduction de la ponctuation par les romains et celle de XML en 1998 par le W3C, il s'était passé pas mal de choses et l'ISO avait notamment, dans les années 80, standardisé le langage de balisage SGML qui permettait déjà de donner aux programmes l'accès aux documents.

Adapté à une utilisation sur le Web

SGML avait été standardisé depuis dix ans lorsque les travaux sur XML ont débuté au W3C en 96 pour en définir un sous ensemble mieux adapté à une utilisation sur le Web.

Impliquant d'avantage les utilisateurs

L'analyse qui a suscité la création de ce groupe de travail est que le relatif échec de SGML qui n'avait pas réussi à sortir de la niche des système de GED était dû à sa complexité et que cette complexité était due en majeure partie à la capacité de SGML de ménager ses utilisateurs.

Si l'on revient à notre exemple, l'approche SGML aurait pu être de définir dans une « DTD » (Document Type Definition) la structure du texte original pour apprendre aux applications à l'interpréter en minimisant à l'extrême le balisage que les utilisateurs doivent rajouter.

XML impose au contraire que tout le balisage soit effectué de manière explicite dans le document.

Plus simple à mettre en oeuvre

Le besoin de développer une DTD pour chaque application SGML et la complexité de mise en oeuvre qu'il entraîne a été perçu comme un des principaux freins à une utilisation sur le Web.

La solution radicale adoptée par XML qui impose que le balisage soit explicite permet d'écrire des documents XML sans DTD et donc sans aucune phase de modélisation préalable.

Simplifiant les problèmes d'échange

Le fait que la plupart des documents SGML ne puissent pas être lus par une application sans connaître la DTD associée complexifie les problèmes d'échanges entre systèmes : il ne suffit pas d'échanger un document mais il faut également échanger ou référencer la DTD associée avec tous les problèmes que cela peut poser en terme de maintenabilité.

L'autonomie des documents XML qui peuvent être lus sans DTD permet au contraire de les échanger et de les archiver beaucoup plus simplement.

EXPOSER LA STRUCTURE, ET APRÈS ?

La galaxie XML

Le fait de connaître la structure d'un document ne permet pas pour autant de le comprendre ni de manipuler les informations qu'il contient. Lorsque l'on parle de XML, on a pris l'habitude d'inclure non seulement le langage de balisage mais également tout une panoplie d'applications et outils que l'on peut classer de la manière suivante :

Identification des vocabulaires, modélisation, validation

Lorsque je transmets le document XML contenant mon accusé de commande, il est utile que je puisse indiquer quel vocabulaire XML est utilisé. A partir de cette identification, il sera également utile que mes interlocuteurs puissent accéder à une modélisation de la structure du document permettant notamment sa validation.

Manipulation des documents

Un standard sans outils ne serait pas quelque chose de bien intéressant et si nous utilisons des standards tels que XML, c'est parce que nous savons que nous pouvons compter sur les outils qui nous permettent de manipuler les documents XML et notamment de les lire et de les écrire en utilisant nos langages de programmation favoris ou des langages spécifiques mieux adaptés à cet usage.

Bibliothèques de vocabulaires

Enfin, pour éviter de réinventer la roue, il est fondamental de disposer de bibliothèques de vocabulaires XML que nous pouvons réutiliser pour constituer tout ou partie de nos documents XML.

IDENTIFICATION, MODÉLISATION, VALIDATION

Identification : espaces de noms

L'identification des vocabulaires auxquels appartient chaque élément et attribut d'un document XML est réalisé au moyen des « espaces de noms ».

Eux mêmes identifiés par des URI

Les espaces de noms XML sont identifiés par des URIs, tels que :

```
http://www.w3.org/1999/xlink
```

ou

```
urn:oasis:names:tc:SAML:1.0:action:ghpp
```

Dans les deux cas, ces URIs ne sont que des identifiants et il n'est pas nécessaire qu'il y ait une page web « à l'autre bout » d'une URL utilisée pour identifier un espace de nom XML.

Représentés localement par des préfixes

Pour éviter d'avoir à répéter ces URIs, on leur affecte des préfixes :

```
<my:Element  
  xmlns:xlink="http://www.w3.org/1999/xlink"  
  xmlns:my="http://example.com/">  
  ...
```

Que l'on utilise ensuite pour y faire référence :

```
<my:crossReference  
  xlink:type="simple"  
  xlink:href="students.xml"  
  xlink:role="http://www.example.com/linkprops/studentlist"  
  xlink:title="Student List"  
  xlink:show="new"  
  xlink:actuate="onRequest">
```

Current List of Students

```
</my:crossReference>  
</my:Element>
```

Ou un espace de noms par défaut

Pour identifier l'espace de noms des éléments, il est également possible de définir un « espace de noms par défaut » :

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" >
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <p>Moved to <a href="http://vlib.org/">vlib.org</a>.</p>
  </body>
</html>
```

Validation : langages de schéma XML

Le domaine de la validation des documents XML est plus varié qu'il n'y paraît à première vue. Lorsque l'on cherche à valider un document XML, on peut en effet valider des aspects aussi divers que :

Validation des structures

La tâche la plus courante dans le domaine de la validation des documents XML est de valider la structure des documents, c'est à dire l'imbrication des éléments et attributs.

Dans le cas de notre commande, on vérifiera ainsi que la commande contient bien un numéro de commande, une date d'expédition, ...

Validation des contenus unitaires (types de données)

Lorsque la structure est validée, on pourra contrôler le type des éléments et attributs en vérifiant de manière individuelle que leurs valeurs correspondent à ce qui est attendu. On vérifiera par exemple qu'une quantité est un entier positif, qu'une date est une date valide, ...

Contrôles d'intégrité référentielle

Il est fréquent de définir dans un document XML des identifiants et de faire ensuite référence à ces identifiants. Les contrôles d'intégrité référentielle vérifient l'unicité des identifiants et l'existence des identifiants auxquels il est fait référence.

Règles métier

On englobe fréquemment tous les contrôles qui ne rentrent pas dans le cadre des catégories déjà citées sous le terme de « règles métier ». Cela englobe des choses aussi variées que de vérifier qu'une date de début est antérieure à une date de fin, qu'un total est égal à la somme des termes ou que l'orthographe d'un document est correcte!

Schémas : deux propositions concurrentes

Combiné multi-fonctions (W3C XML Schema)

Le langage de schéma W3C XML Schema tente de couvrir un maximum de terrain (c'est à dire la validation des structures, celle des types de données et les contrôles d'intégrité référentielle) au prix d'une grande complexité.

Boite à outils (ISO DSDL, RELAX NG, Schematron, ...)

Le projet ISO/DSDL (<http://dSDL.org>) vise au contraire à définir une boite à outils composé de langages simples et spécialisés assurant chacun une fonction de manière optimale.

Les plus connus de ces langages sont RELAX NG (validation de structure) et Schematron (règles métier).

Modélisation

Il n'y a pas vraiment de standardisation dans la manière de modéliser des documents XML et différentes approches peuvent coexister.

UML

UML peut parfaitement être utilisé pour modéliser des documents XML. Complété par des stéréotypes adaptés, un schéma UML permet de générer la documentation et les schémas XML assurant la validation des documents.

Schémas XML

Parmi les catégories de langages de schéma proposées ci-dessus, les schémas permettant de valider la structure des documents et les types de données peuvent être considérés comme des modélisations des documents XML.

Nomenclatures

Une simple nomenclature définissant la liste des éléments et des attributs ainsi que leur contenu et types de données peut également constituer la modélisation d'un vocabulaire XML. Si cette liste est établie de manière structurée (par exemple en XML ou en utilisant le format XML d'un tableur tel que Excel, OpenOffice ou Gnumeric), il est facile de générer le schéma correspondant.

Échantillons de documents

Les échantillons de documents sont généralement plus faciles à lire que les schémas correspondants et on peut utiliser ces échantillons comme modélisation et générer les schémas correspondants.

MANIPULATION DE DOCUMENTS XML

Parseurs XML

Les parseurs XML sont des interfaces permettent aux applications d'accéder aux informations présentes dans un document XML. On distingue deux grandes familles de parseurs XML.

Parseurs orientés flux (SAX)

Les parseurs « orientés flux » dont SAX est le principal représentant communiquent les informations qu'ils trouvent dans les documents XML au fur et à mesure de leur analyse syntaxique en ne mémorisant que le contexte nécessaire à vérifier que le document est « bien formé » et éventuellement valide suivant un schéma XML.

Ils sont économes en consommation mémoire et peuvent être utilisés de manière efficace pour constituer des « pipes » de transformations de documents XML.

Parseurs orientés modèle (DOM)

Les parseurs « orientés modèle » dont DOM est le principal représentant lisent les documents XML et les stockent sous forme de modèles orientés objet qui sont transmis à l'application pour lecture et modification.

Plus gourmands en mémoire, ils permettent aux applications d'avoir facilement accès à l'ensemble des informations présentes dans les documents.

Binding XML/objets

L'utilisation de parseurs est laborieuse

Sans être complexe, l'utilisation d'un parseur SAX ou DOM pour alimenter une structure d'objets demande une charge de travail relativement importante et ce travail est source d'erreurs.

Automatiser le « binding » entre XML et objets

Il est donc naturel de chercher à automatiser tout ou partie du binding entre documents XML et systèmes d'objets. De nombreuses propositions ont été faites pour cela et je pense que c'est un domaine dans lequel de gros gains de productivité pourraient être réalisés.

Approches binding XML/objets

Création de classes à partir de schémas (Java JAXB, .Net, ...)

L'approche la plus répandue pour les langages de programmation à typage statique tels que Java ou C# est la création de classes à partir des schémas XML.

C'est une approche qui convient bien aux développements de nouvelles applications à partir de la définition d'un vocabulaire XML exprimé sous forme de schema XML.

Dans l'univers Java, un tel système a été standardisé en « JSR » sous le nom de JAXB et un autre est disponible en standard en environnement .Net.

Génération dynamique de classes à partir de documents (langages à typage dynamique)

Dans le cas des langages à typage dynamiques (tels que Perl, Python, Rubis, PHP, ...) il est possible de générer dynamiquement les classes à partir des documents XML et d'accéder aux informations du document en utilisant la syntaxe classique du langage utilisé. De nombreuses bibliothèques permettent de réaliser cela dont celle qui fera l'objet de ma présentation « XML Driven Classes in Python » le 29 juillet à Portland (http://conferences.oreillynet.com/cs/os2004/view/e_sess/5058).

Binding dynamique par introspection

Dans le cas où des classes métier existantes doivent être utilisées et où la structure des documents correspondant est voisine de celle de ces classes, il est également possible de réaliser le binding par introspection.

Moins répandue, cette approche me semble néanmoins très prometteuse et je suis en train de développer une bibliothèque suivant ce principe dans le cadre d'un projet pour l'INSEE.

J'ai l'intention de publier cette bibliothèque sous licence Open Source, n'hésitez pas à me contacter si vous êtes intéressé par son utilisation.

Support direct par les langages

Une voie à tracer

Compte tenu de la généralisation de l'utilisation de XML, il devient souhaitable que les langages de programmation considèrent les fragments de documents XML comme des types natifs et offrent ainsi un accès direct aux informations qu'ils contiennent.

Un proposition en ce sens à été fait lors de XML 2003 pour C# par Erik Meijer (<http://research.microsoft.com/~emeijer/Papers/XML2003/xml2003.html>) et c'est

également un domaine porteur de gains de productivité importants dans le domaine de la programmation XML en général et des Services Web en particulier.

Langages spécialisés

XSLT 1.0 (transformations, typage dynamique)

XSLT 1.0 est un langage de programmation à part entière. Il est orienté « transformations » et une transformation XSLT est constitué d'un ensemble de règles (« templates ») qui définissent les traitements à effectuer en fonction des noeuds rencontré lors du parcours d'un document XML.

XSLT 1.0 est un langage à typage dynamique: la manipulation des documents ne fait appel à aucun schéma et les variables et paramètres sont typés en fonction des valeurs qui y sont placées.

XSLT 1.0 est une recommandation W3C publiée en novembre 1999.

XSLT 2.0 (transformations, typage statique)

Si le principe de base de XSLT 2.0 reste le même, XSLT 2.0 devient un langage à typage statique et pour bénéficier de toutes ses fonctionnalités, il faut associer des schémas aux documents manipulés et déclarer le type des variables et paramètres.

XSLT 2.0 est une spécification W3C actuellement à l'état de version de travail (« Working Draft »).

XQuery (généraliste, typage statique)

Spécifié par le même groupe de travail que XSLT 2.0, XQuery est, contrairement à ce que son nom laisse supposer, un langage de programmation beaucoup plus générique que XSLT 2.0.

Il n'est pas gouverné par l'approche déclarative de XSLT et adopte une approche « programmatique » beaucoup plus classique.

C'est un langage à typage statique au même sens que XSLT 2.0 et c'est également une spécification W3C à l'état de version de travail.

BIBLIOTHÈQUES DE VOCABULAIRES

Absence de centralisation

La principale caractéristique de la recommandation des espaces de noms XML est de refuser toute tentation de centralisation en ce qui concerne l'identification des espaces de

noms et donc des vocabulaires XML.

Cela se traduit par le fait qu'il n'y a aucun répertoire central dans lequel figurent tous les vocabulaires ou espaces de noms XML.

Vocabulaires horizontaux ou verticaux

Parmi les vocabulaires publiés, on retrouve l'éternelle classification entre vocabulaires horizontaux ou génériques et logiciels verticaux ou métier.

Vocabulaires horizontaux : prédominance du W3C (Xlink, Xinclude, XML Signature, XML Encryption, RDF, ...)

Le W3C est très actif dans le domaine des logiciels horizontaux et l'on citera notamment XLink (expression de liens), XInclude (inclusion de documents), XML Signature (signature de fragments de documents), XML Encryption (chiffrement de fragments de documents), RDF (expression de graphes en XML), ...

Vocabulaires verticaux : grande dispersion (W3C pour le Web, Oasis, UN/CEFACT, ISO...)

En ce qui concerne les vocabulaires plus « verticaux », on assiste à une très grande dispersion.

Le W3C est actif dans les domaines touchant au Web (XHTML, SVG, Smil, XForms, ...). Oasis intervient dans les domaines les plus variés, parfois en association avec l'UN/CEFACT (projet ebXML). L'ISO a tendance à « xmliser » ses normes existantes et à utiliser XML pour ses nouveaux travaux.

A côté de ces poids lourds de la standardisation, de nombreux groupements existants ou créés pour l'occasion publient des définitions de vocabulaires XML.

Répertoire des schémas XML de l'administration

En France, il faut également citer le répertoire des schémas XML de l'administration (http://www.adae.gouv.fr/article.php3?id_article=167) géré par l'Agence pour le Développement de l'Administration Électronique (ADAÉ).