



Dix années de XML

Répondez à cet article.

La recommandation XML 1.0 a été publiée il y a exactement dix ans jour pour jour. C'est l'occasion de revenir sur ce qu'a été XML pour moi pendant ces dix années.

Eric van der Vlist, Dyomedea (vdv@dyomedea.com).
dimanche 10 février 2008

Table des matières

[Découverte de XML](#)

[XML pour le Web](#)

[RSS 1.0](#)

[DSDL](#)

[Recherche de perfection](#)

[XML sur le Web](#)

[XML est un état d'esprit](#)

[XML et mondialisation](#)

Découverte de XML

Au risque de vous décevoir; je dois avouer que la publication de la recommandation **XML** 1.0 le 10 février 1998 est passée totalement inaperçue pour moi et que je n'avais à l'époque aucune idée de ce qu'était **XML**.

J'étais responsable du support européen de deuxième niveau pour la gamme des produits serveurs de **Sybase** et mon activité ne me donnait pas beaucoup de raisons de m'intéresser à **XML**. J'étais par contre un fervent utilisateur du Web, en externe et

Mots clés.

- [Aaron Swartz](#)
- [Apple](#)
- [Bloglines](#)
- [CGI](#)
- [Charles Goldfarb](#)
- [Charles Golfarb](#)
- [Dave Winer](#)
- [DOM](#)
- [DSDL](#)
- [Du côté de...](#)
- [Edd Dumbill](#)
- [Eric van der Vlist](#)
- [GML](#)
- [Google](#)
- [Google Reader](#)
- [HTML](#)
- [IEC](#)
- [ISO](#)
- [James Clark](#)
- [Java](#)
- [Jon Bosak](#)
- [JSON](#)
- [Linux](#)
- [Manuel Lemos](#)
- [MCF](#)
- [Meerkat](#)
- [Murata Makoto](#)
- [MySQL](#)
- [NetObjects Fusion](#)
- [Netscape](#)
- [Netvibes](#)
- [NRL](#)
- [O'Reilly](#)

surtout en interne : les sites Intranet poussaient comme de la mauvaise herbe et le support technique n'était pas en reste.

J'avais commencé à développer mes premiers sites dynamiques en **shell** script puis en **Perl** en **CGI** et à grand renfort de « print » de morceaux de **HTML** avant de me laisser tenter par **Java**.

Pour éviter de lancer une machine virtuelle à chaque requête, j'avais développé quelque chose qui ressemble beaucoup à ce que nous appelons maintenant un serveur de servlets : une machine virtuelle multithread tournant en tâche de fond et communiquant via des sockets **TCP/IP**. Et pour éviter les « print » de fragments **HTML**, j'avais également développé une bibliothèque représentant les éléments **HTML** que j'utilisais sous forme de classes **Java** : mon application créait un modèle du document à envoyer qui ressemblait fort à ce que nous appelons un **DOM** et sérialisait le tout avant de l'envoyer au client.

Fort de cette boîte à outils, j'avais écrit plusieurs applications s'interfaisant avec la base de données de nos systèmes de gestion des incidents et des bugs et cela simplifiait la gestion de mon équipe et la communication avec l'engineering et le support technique de premier niveau.

Tout cela m'avait donné suffisamment confiance en mes capacités de développeur Web pour lancer un site après mon départ de **Sybase**.

« Du côté de... » était un site de proximité qui voulait utiliser Internet pour rapprocher les habitants de mon quartier des ressources locales. J'avais une nouvelle fois changé de langage de programmation pour utiliser **PHP** et troqué **Solaris** que j'utilisais chez **Sybase** contre **Linux**.

Pour des raisons de coûts, le site était hébergé chez un petit hébergeur situé aux États-Unis et sa fiabilité n'était pas toujours à la hauteur de mes espérances. Pour mieux surveiller l'état de mon serveur, je lançais sur la liste de discussions « php-users » l'idée d'un système communautaire et distribué utilisant les serveurs de ses membres pour se surveiller mutuellement.

Manuel Lemos se montra intéressé et nous commençâmes à échanger des messages pour définir l'architecture du projet baptisé « Web Watchers ». Mon premier message en esquissa les grandes lignes :

- [P3](#)
- [Perl](#)
- [PH](#)
- [PHP](#)
- [PHP3](#)
- [Planet XMLhack](#)
- [PostgreSQL](#)
- [Rael Dornfest](#)
- [Ramanathan V. Guha](#)
- [RDF](#)
- [RELAX](#)
- [RELAX NG](#)
- [Rick Jelliffe](#)
- [RSS](#)
- [SaaS](#)
- [Schematron](#)
- [SGML](#)
- [shell](#)
- [Solaris](#)
- [SQL](#)
- [Sybase](#)
- [TCP/IP](#)
- [TREX](#)
- [URI](#)
- [W3C XML Schema](#)
- [Web](#)
- [Web 2.0](#)
- [XHTML](#)
- [XML](#)
- [XML Europe](#)
- [XMLfr](#)
- [XP](#)
- [XPath](#)
- [XSLT](#)

ACTUALITÉ

- [stcML 1.0](#)
- [Dix années de XML à l'ABES](#)
- [Dix années de XML chez Adobe France](#)
- [Dix années de XML à l'INSEE](#)
- [Dix années de XML](#)
- [XML 1.0 5ème édition](#)

"Je pense à un daemon écrit en **Perl** tournant sur les machines concernées et contrôlé via des pages **PHP3** (ce serait très portable et cela signifie probablement que l'on utiliserait des fichiers de configuration plutôt qu'une base de données quelconque)."

C'est sa réponse qui me révéla l'existence de **XML**, le 16 juin 1999 et je lui en suis toujours reconnaissant :

"Oui, si ces fichiers ne sont pas destinés à être modifiés lors de l'exécution, nous pourrions utiliser des fichiers **XML** parce qu'ils sont faciles à modifier à la main et à parser en **PHP**."

Peu convaincu par la facilité d'utilisation du parseur **XP** interfacé à **PHP3**, je développais pour ce projet une bibliothèque **PHP**, [xmltree.inc](#) qui assurait le binding d'un document **XML** dans une structure d'objets **PHP** créée dynamiquement. C'était en quelque sorte un des premiers systèmes de binding entre **XML** et des objets.

Cette classe eut un certain succès et fut adoptée par [Rael Dornfest](#) pour l'agrégateur de flux **RSS** « [Meerkat](#) » qu'il avait développé pour **O'Reilly**.

XML pour le Web

Web Watchers ne vit jamais le jour faute de temps, mais mon intérêt avait été éveillé.

Du côté de... était un site dynamique bâti sur **PostgreSQL** (mon passé au support technique chez **Sybase** m'avait fait rejeter comme un gadget **MySQL** qui ne gérait pas la notion de transactions). Une base de données **SQL** convenait très bien pour gérer la plupart des fonctionnalités du site, mais sa structure très régulière manquait de souplesse pour personnaliser les pages que nous appelions les « [vitrines](#) des commerçants ».

Ces vitrines étaient vendues aux commerçants du quartier et elles cherchaient à être à la fois homogènes pour que les visiteurs trouvent facilement les informations communes à chaque vitrine et personnalisées. Je prenais des photos des boutiques et avec l'aide de chacun d'entre eux, je réalisais un montage spécifique destiné à traduire au mieux les particularités de leur commerces.

La description de ces montages était spécifique à chaque vitrine

- [HTML 5, un grand pas...](#)
- [Huit bougies pour XMLfr](#)
- [Planète XMLfr](#)
- [XML 2006 : souvenirs, souvenirs](#)
- [Web 2.0 : risques et perspectives](#)
- [Une projection qui simplifie RDF](#)
- [Parser un nœud texte en XML](#)
- [Version XML/RDF/OWL du COG](#)
- [Créer une table des matières en XSLT](#)
- [Contrat de Cyber Embauche \(CCE\)](#)
- [Services Web : de l'infrastructure à la sémantique](#)
- [L'internaute 2.0 est-il un animal social ?](#)
- [Attributs xml et schémas W3C XML Schema](#)
- [Remote Events for XML \(REX\)](#)

et c'est pour gérer cette diversité que j'utilisais mes compétences **XML** fraîchement acquises. Je décidai donc d'écrire la description de ces vitrines en XML et de générer les pages **PHP** correspondantes à partir de cette description. Et pour cette génération, plutôt que d'utiliser un langage généraliste, je décidai d'essayer **XSLT** qui était encore en cours de normalisation.

Un peu plus tard, à l'automne 1999, j'entrepris également la migration du [site de ma société](#) développé avec **NetObjects Fusion** que je trouvais trop contraignant et adoptais un mécanisme de génération de pages statiques en **XSLT** à partir de leur description en **XML**.

Lorsque fin 1999 j'ai du me rendre à l'évidence et admettre que **Du côté de...** malgré le succès qu'il avait auprès de ses visiteurs ne serait jamais économiquement viable, je maîtrisais suffisamment bien les quelques recommandations **XML** publiées à cette époque (**XML**, espaces de noms, **DOM** niveau 1, **XPath** 1.0 et **XSLT** 1.0) pour pouvoir me positionner comme expert **XML** et décidai de lancer **XMLfr** comme je l'ai déjà expliqué [ailleurs](#).

Les choses s'enchaînèrent ensuite très vite. La visibilité que me donna **XMLfr** me permit de trouver des contrats de conseil et de formation qui me permirent à leur tour d'accroître mon expérience dans la mise en œuvre des technologies **XML**. En parallèle, **XMLfr** me donna un prétexte pour faire, sous couvert de journalisme, beaucoup de veille technologique et de conserver ainsi une bonne vision technique.

Je ne vais pas décrire ici huit années d'expérience dont la majeure partie est directement liée à **XML** mais plutôt revenir sur deux moments forts.

RSS 1.0

Le premier est, entre juin et décembre 2000, ma participation aux travaux qui aboutirent à la spécification **RSS** 1.0.

C'est Rael Dornfest qui m'entraîna dans cette aventure en m'expliquant à quoi il utilisait ma bibliothèque **PHP** `xmltree.inc` dans son agrégateur de flux **Meerkat**. **Meerkat** est en quelque sorte l'ancêtre des agrégateurs de flux tels que nous les connaissons aujourd'hui. A la différence d'un **Bloglines**, **Google Reader** ou autre **Netvibes**, **Meerkat** avait sa propre liste de flux **RSS**, tous relatifs à l'actualité du développement logiciel au

sens large et classé en catégories. Mi 2000, cette application était réellement innovante et faisait de Rael un des pionniers de l'utilisation de **RSS** en tant que service « consommateur » de flux.

Rael m'entraîna sur la liste de discussion syndication@egroups.com sur laquelle il débattait des problèmes liés à l'extension de **RSS** et je défendis l'idée de revenir à une utilisation de **RDF** en réponse à une de ses propositions pour ajouter des fonctionnalités de gestion de fils de discussions :

"Cette fonctionnalité est vraiment sympa, mais ne devrait-on pas envisager d'utiliser **RDF**? J'ai bien peur que nous ne réinventions le triplet **RDF** ;=) !"

Il fut bientôt évident que l'évolution de **RSS** était bloquée par la volonté de Dave Winer opposé à la notion de modularité et étendant conserver une spécification monolithique sur laquelle il continuerait à contrôler toute nouvelle fonctionnalité et mis au défi de créer une nouvelle branche, nous nous mirent au travail sur la liste de discussion rss-dev@egroups.com. Ce fut une aventure passionnante que vous pouvez revivre sur les archives de cette liste de discussion. Tout était à créer. La spécification bien entendu, mais également l'équipe et les procédures avec lesquelles elle fonctionnerait.

Travailler ainsi avec une dizaine de personnes que l'on a, pour la plupart, jamais rencontré peut réservier des surprises. Un de nos membres les plus actifs s'appelait [Aaron Swartz](#). Ses contributions étaient généralement pleines de bon sens et j'étais certain qu'elles traduisaient une longue expérience en développement logiciel. Ce n'est que lorsque je lui demandais pourquoi il ne viendrait pas à une des conférences à laquelle je participais et qu'il me répondit que ses parents ne lui autoriseraient pas à manquer ses cours que j'appris qu'il n'avait que quatorze ans!

RSS 1.0 était aussi une des premières applications de **RDF** pouvant toucher le grand public. Nous devions donc travailler à deux niveaux et avec deux communautés utilisant des outils différents : d'un côté **RSS 1.0** devait rester un vocabulaire **XML** aussi simple que possible et facile à utiliser avec des outils **XML** et de l'autre il devait également respecter la syntaxe de **RDF** et avoir un modèle de données qui tienne la route lorsque ses documents sont découpés en triplets.

Nous avions également parmi notre équipe les inventeurs de

RSS et de ses ancêtres. C'est le cas de [Ramanathan V. Guha](#), qui avait développé chez **Apple** le format **MCF** (Meta Content Framework) avant de rejoindre **Netscape** pour en dériver un format **XML/RDF** publié sous le nom de **RSS** 0.9...

Ce sont ces anciens qui insistèrent pour que nous gardions le nom de **RSS** lorsque ce délicat problème commença à se poser, ce qui fut sans doute notre plus grosse erreur puisque **RSS** 1.0 apparut comme une version obsolète lorsque Dave Winer décida de publier **RSS** 2.0.

Nous étions tous convaincu du potentiel du concept de « syndication » et je me souviens que nous évoquions déjà la possibilité d'utiliser nos flux pour décrire des documents audio et vidéo, pratique que nous appelons maintenant podcast...

C'est en discutant de ce type d'applications qui semblaient encore bien lointaines que je pris conscience de la nécessité de trouver le bon niveau de détail lorsque l'on écrit une spécification. J'avais auparavant tendance à penser qu'une bonne spécification doit chercher à tout décrire et mon expérience dans le groupe de travail **RSS** 1.0 m'a permis de comprendre que la « sur-spécification » est aussi dangereuse que la « sous-spécification ».

La « sur-spécification » conduit à des discussions interminables et allonge le temps de développement des spécifications, mais ce n'est pas son défaut le plus grave.

J'ai ainsi cherché, pendant un moment, à définir ce qu'était un « item » **RSS** avant de prendre conscience que toute définition trop précise risquait d'être restrictive : spécifier qu'un item était une page web interdisait d'utiliser un item pour décrire une photo ou un document multimédia. Pour préserver l'avenir, il fallait donc qu'un item reste quelque chose d'aussi vague que possible et nous avons au contraire pris la [précaution](#) de spécifier qu'un item n'était pas nécessairement ce que l'on avait l'habitude de voir dans un flux **RSS** :

"Bien que ce soit généralement une nouvelle, compte tenu de l'extensibilité de RSS 1.0, ce [NDT: un item] peut être à peine près n'importe quoi : un message dans une discussion; une offre d'emploi, un patch logiciel : n'importe quel objet qui a un **URI**."

Cette loi s'applique de manière générale à la spécification de modèles ou de formats de données. Elle ne s'applique pas de la même manière à la spécification de langages de programmation

tels que **XSLT** ou **XPath** ou de schémas qui demandent au contraire une grande précision, mais c'est une des grandes forces de la recommandation **XML** 1.0 elle même de laisser des zones d'ombres.

Ce sont ces zones d'ombre qui ont permis d'une part de publier une recommandation en un temps record (moins de deux ans) et d'autre part d'utiliser **XML** de manière large sans cantonner son utilisation à un domaine d'application ou à un environnement particulier.

DSDL

Le deuxième de ces moments forts est ma participation au groupe de travail **ISO-DSDL** et plus particulièrement les séances de travail de mai 2002 à l'occasion de **XML Europe 2002**.

XML Europe 2002 reste pour moi le point culminant en matière de créativité technique dans les technologies **XML** et c'est la conférence pendant laquelle j'ai suivi le plus de sessions qui m'ont donné à réfléchir. La plupart des grandes spécifications étant publiées, un certain nombre de personnes ont pu avoir tendance à considérer que les grands travaux autour de **XML** étaient terminés et qu'ils pouvaient passer à autre chose et délaisser ce type de conférence.

C'est en tout cas à **XML Europe** 2002 que j'ai pu rencontrer le groupe de travail **ISO-DSDL**, véritable dream team **XML** dans laquelle se trouvaient réunis James Clark, Murata Makoto et Rick Jelliffe sous la présidence de Charles Goldfarb.

Charles Goldfarb est le « G » de **GML** et le principal créateur de **SGML**. Bien qu'il ait été très en retrait par rapport aux travaux techniques de notre groupe de travail, il présidait encore le groupe « **ISO/IEC** JTC 1/SC 34 WG 1 » auquel nous étions rattaché. J'ai eu l'occasion de déjeuner plusieurs fois avec lui et se fut toujours un plaisir de discuter avec ce précurseur qui semblait comprendre à mi-mots les problèmes qui me passionnaient.

De l'avis de Jon Bosak qui a présidé le groupe de travail **XML**, James Clark a été le responsable technique de la recommandation **XML** 1.0. Il a ensuite rédigé les recommandations **XPath** et **XSLT** 1.0 avant de s'intéresser aux langages de schémas et de travailler sur **TREX**, **RELAX NG** et **NRL**. Sa contribution aux technologies **XML** est réellement capitale et je le tiens James pour le plus brillant des experts

XML avec lesquels j'ai pu travailler.

Moins connu et moins médiatique que James Clark, Murata Makoto est l'auteur de **RELAX**; un langage de schéma qui a été fusionné avec **TREX** pour produire **RELAX NG**. Plus encore que James Clark, c'est lui qui est à l'origine de la solide base mathématique de **RELAX NG**. James Clark et Murata Makoto se sont complétés de manière admirable lors de la création de **RELAX NG** et ont fourni des implémentations utilisant deux algorithmes différents démontrant ainsi que la plupart des restrictions que s'imposent **W3C XML Schema** de crainte à être difficilement implantables ne sont pas justifiées.

Moins théoricien et plus pragmatique, Rick Jelliffe est peut-être le plus innovant et imprévisible des trois. Trouvant toujours des manières innovantes et étonnantes d'utiliser des technologies existantes, c'est l'inventeur de **Schematron**, un langage de schéma simple qui exprime des contraintes sous forme d'expressions **XPath**. Plutôt que de réinventer quelque chose de nouveau, **Schematron** est en quelque sorte une astuce géniale qui permet d'utiliser **XPath** comme un langage de schémas.

Ma contribution au groupe de travail **DSDL** ne s'est pas limitée aux réunions que nous avons eu en marge de **XML Europe** 2002 mais ce fut la principale occasion pendant laquelle l'équipe fut réunie physiquement et la luminosité que dégage encore ce souvenir n'est pas uniquement due au soleil de Barcelone!

Travailler avec eux fut d'autant plus un plaisir qu'ils sont toujours restés très ouverts aux critiques. Les occasions de ne pas être d'accord avec eux étaient rares, mais je me souviens notamment avoir été en désaccord avec James Clark sur l'utilisation de noms qualifiés par des préfixes d'espaces de noms dans des attributs de la syntaxe **XML** de **RELAX NG** et avoir obtenu partiellement gain de cause puisqu'il est possible d'utiliser **RELAX NG** sans utiliser ces noms qualifiés.

J'ai beaucoup appris de ces deux expériences complémentaires et ces expériences me sont encore précieuses lorsqu'il s'agit de concevoir et de définir des vocabulaires **XML** ce qui représente une part importante de mon activité de conseil.

Recherche de perfection

L'approche de James Clark et Murata Makoto qui ont créé **RELAX NG** en réponse aux imperfections de **W3C XML Schema** est particulièrement exemplaire puisqu'elle démontre que nous

ne sommes pas condamnés à utiliser des outils médiocres parce qu'ils sont standards mais qu'au contraire, une certaine forme de recherche de perfection est possible.

C'est d'autant plus important qu'à partir de 1999, un grand nombre de spécifications ont été élaborées en très peu de temps avec des niveaux de qualité très inégaux. C'est en exerçant un regard critique que l'on peut sélectionner celles qui méritent de l'être et chercher de meilleures alternatives à celles qui ne le méritent pas.

C'est cette recherche qui maintient mon intérêt pour les technologies de base **XML**. Alors que beaucoup considèrent que tout a été dit sur le sujet et qu'il faut passer à autre chose pour innover, je pense au contraire qu'il reste encore beaucoup de choses à améliorer y compris même aux niveaux les plus basiques.

XML sur le Web

Il serait d'ailleurs d'autant plus dommage de baisser les bras que contrairement à ce que l'on pourrait penser, le succès de **XML** est fragile et contesté.

La situation est d'autant plus paradoxale que **XML** a été conçu pour « mettre **SGML** sur le **Web** » et que c'est sur le **Web** qu'il est actuellement remis en cause et ce pour les deux types d'applications qui lui semblaient acquis.

Le premier de ces domaines est l'échange de données entre applications où **XML** est de plus en plus controversé et souvent remplacé par **JSON**, au moins en ce qui concerne les applications **Web 2.0**. Le second est le format de publication des pages **Web** elles mêmes où **HTML 5** et son format non **XML** vient fragiliser **XHTML** vraisemblablement condamné à rester marginal sur le **Web**.

Dans les deux cas, les détracteurs de **XML** s'appuient sur les défauts des spécifications complémentaires à **XML** (notamment les espaces de noms, le langage **W3C XML Schema**, l'API **DOM**, ...) beaucoup plus que sur les défauts de **XML** en tant que tel.

XML est donc victime d'un certain laxisme qui a conduit à accepter des imperfections jugées mineures par ses défenseurs et qui sont autant de prises pour ses détracteurs.

Pourquoi ce rejet de **XML**? Je pense qu'il peut être pris à deux

niveaux.

A un niveau superficiel, on peut ne rejeter que la syntaxe de **XML**. C'est le cas lorsque l'on substitue simplement **JSON**, **HTML** ou tout autre format à **XML**. Ce type de substitution ne remet pas en cause l'architecture et il est toujours facile de convertir ces formats en **XML**.

XML est un état d'esprit

C'est en effet le grand paradoxe de **XML** que bien qu'il ne soit que syntaxe (la recommandation **XML 1.0** ne définit que la syntaxe des documents **XML**) sa syntaxe soit accessoire (il est facile de convertir tout document structuré en **XML**).

Edd Dumbill a donné à **Planet XMLhack** un sous titre que je trouve excellent : « Angle brackets are a way of life » que l'on peut traduire un peu librement par « le balisage est une manière de vivre ». Je pense effectivement que **XML** est avant tout un état d'esprit, une volonté d'ouverture et que le point important est l'échange de documents sous forme de formats ouverts et non la syntaxe de **XML**.

Cet aspect « échange de documents » est lui même violemment rejeté par beaucoup de développeurs. Il faut dire qu'il représente une remise en cause profonde de l'architecture du système d'information. Alors que l'on raisonne encore principalement en terme d'APIs synchrones manipulées par des langages procéduraux, l'approche « orientée documents » privilégie un couplage plus lâche et souvent asynchrone entre applications et les langages déclaratifs lui sont souvent mieux adaptés.

Des changements de cette ampleur risquent d'autant plus d'être rejetés que si l'on confie leur mise en œuvre à des équipes qui ne sont pas motivées et expérimentées dans ce domaine, on se heurte immanquablement à des problèmes techniques. Ces équipes ont ensuite toute latitude pour expliquer que « cela ne marche pas » et qu'il faut revenir à une approche d'APIs classiques.

Ce scénario n'a malheureusement rien d'exceptionnel et n'épargne pas les équipes de développement les plus modernes.

XML et mondialisation

Face à de tels rejets, comment **XML** a-t-il pu prendre l'importance que nous lui connaissons?

Je me souviens avoir suivi une présentation d'un vocabulaire **XML** au cours de laquelle l'orateur expliquait pour introduire **XML** que **XML** était aux données ce que la standardisation des conteneurs avait été au commerce international.

XML est effectivement un standard qui permet de transporter des données dans des conteneurs standards et les forces qui ont assuré le succès de **XML** sont les mêmes que celles qui ont assuré le succès du libre échange des marchandises.

Si des sociétés concurrentes ont décidé il y a dix ans de préférer **XML** aux formats propriétaires qu'elles utilisaient auparavant, c'est qu'elles ont estimé que l'ouverture qui en résulterait serait favorable à leurs intérêts.

Les conséquences de ce libre échange de données sont d'ailleurs similaires au libre échange de biens physiques et c'est ce qui explique que **Google** soit aujourd'hui en mesure de dominer le marché de la publicité sur Internet en France comme ailleurs.

Je ne veux pas dire pour autant qu'il faille se recroqueviller sur soi-même et combattre cette mondialisation des données qui est également un formidable outil d'échange culturel, mais il me semble important d'avoir conscience du fait que ce sont les formats ouverts en général et **XML** en particulier qui en permettent la réalisation technique.

Ce mouvement de mondialisation est en train de franchir une nouvelle étape avec ce que nous appelons maintenant **SaaS** (Software as a Service) qui n'est rien de moins que la mondialisation des applications et qui est elle aussi rendue possible par les formats ouverts et **XML** en particulier.

Cette étape semble désormais certaine et elle risque de trancher de manière brutale le débat entre architecture orientée documents et architecture orientée API au sein des services de développement informatique.

Les forces qui soutiennent l'« état d'esprit **XML** » sont donc toujours bien présentes!

Je n'ai pas de boule de cristal pour prédire ce que sera **XML** dans dix ans, mais je suis certain que le domaine des formats de données ouverts dans l'état d'esprit **XML** (si ce n'est dans la forme que nous lui connaissons aujourd'hui) nous promet encore

des années de travail techniquement intéressant et qu'il continuera à être au cœur des changements qui bouleverseront l'informatique.

XML au sens large n'est pas prêt à céder sa place d'observatoire des révolutions de l'informatique!

Copyright 2008, Eric van der Vlist.

Les documents publiés sur ce site le sont sous licence "[Open Content](#)"



Conception graphique
I.henriot

Conception, réalisation et hébergement

Questions ou commentaires
redacteurs@xmlfr.org