

TIRÈME SARL

Schema, Schéma

Rédaction : Pierre Attar

Recommandation(s) liée(s) : [DSDL](#) - [DTD](#) - [Infoset](#) - [RELAX NG](#) - [XML](#) - [XML Base](#) - [XMI](#) - [XPath](#) - [WSDL](#)

Partant du principe que les [DTD](#) issues de la Recommandation [XML](#) ne permettaient pas de contraindre suffisamment les données et les structures, partant également du principe qu'il n'y avait aucune raison pour que les modèles ne soient pas codés sous forme [XML](#), [Schema](#) propose une méthode de réalisation de modèles, alternative aux [DTD](#).

Cette spécification, plus moderne de conception que les [DTD](#), permet d'écrire des modèles de façon plus efficace. De fait, et dans son utilisation, cette spécification supprime maintenant l'utilisation des [DTD](#) et, hors environnement contractuel spécifique, il reste peu d'endroits où de nouveaux projets utilisent des [DTD](#).

L'avenir de cette spécification ? Trouver une intégration harmonieuse avec la spécification [DSDL \(RELAX NG\)](#) qui a un champ de recouvrement important (voir par exemple la dernière version de [DocBook](#) qui préfère se reposer sur [RELAX NG](#) plutôt que d'utiliser les [DTD](#) ou les [Schema](#).

Objectifs

L'objectif des [Schema](#) est de définir des contraintes sur des classes de documents conformes à un même modèle. À la différence des [DTD](#), qui ne définissaient *que* les relations entre les différents composants d'un document, les [Schema](#) peuvent typer les données et, par là-même, en documenter leur sens et, donc, leur utilisation.

L'objectif n'est pas d'avoir un langage extensible permettant d'exprimer n'importe quelle contrainte : seules, les contraintes consensuelles sont exprimables. Celles-ci doivent pouvoir être validées par un *parser* et être mises en jeu pour assister à la création d'information, surtout dans les éditeurs.

Enfin, les [Schema](#) ne s'intéressent pas à la validation d'un document [XML](#) (d'un fichier) en tant que tel. Un outil validant se basera sur la représentation en arbre d'objets typés et *valués*, désérialisée selon le standard [Infoset](#). Cette volonté permet de rendre le processus de validation indépendant d'une syntaxe de codage quelconque d'un document.

Principes

Les spécifications sur les [Schema](#) sont divisées en deux parties : une sur les *structures* et une sur les *types de données*. Pour mieux comprendre ces deux recommandations, une introduction est aussi proposée.

D'un point de vue structurel, un **Schema** se définit comme étant l'assemblage, sous forme d'arbre **XML**, d'un ensemble de composants. Il en existe 12, dont les principaux sont :

composants de définition de types, les simples ou les complexes :

ils permettent de typer des éléments pour leur affecter un modèle de contenu. Il en va de même pour les attributs. Ainsi, une *adresse de livraison* et une *adresse de facturation* relèveront toutes deux d'un même type *adresse*.

C'est en utilisant les types simples que seront définis, dans la partie "*datatypes*" (), des types de base comme les chaînes de caractères, les nombres, les **URI** ou encore toutes sortes de dates.

Une définition de "type" peut être réalisée en *restriction* ou en *extension* d'une définition existante. Par exemple, le type *adresse* précédent peut être restreint à des adresses françaises, en jouant sur la définition de la valeur du code pays. Une limite du mécanisme d'extension est que l'on ne peut rajouter des éléments *qu'à la fin* d'un type pré-existant et que l'on ne peut pas les modifier.

D'un point de vue programmation, les notions de types de données permettent de se reposer sur ces types de données pour effectuer des traitements partagés à différents éléments du même type. Ceci simplifie beaucoup les méthodes de programmation, tout en nécessitant d'avoir toujours accès au modèle, en même temps que l'on effectue les traitements.

composants de déclaration : éléments, attributs et notations

Ces composants permettent de déclarer, comme avec les **DTD**, des éléments, avec leurs modèles de contenus, et ainsi que des attributs. Pour ce faire, et dans le but de factorisation précédemment expliqué, les déclarations peuvent se reposer sur des composants de définition de type.

composants groupes : pour la factorisation d'informations de modèles de contenus ou d'attributs.

Ces composants permettent de définir des modèles qui seront souvent réutilisés, soit pour définir des composants de déclaration, soit pour définir des composants de définition de types. Dans les **DTD** traditionnelles, ces groupes étaient formalisés par des entités paramètres, qui permettaient, par exemple, de définir tous les composants *blocks* (paragraphes, listes, remarques, etc.) nécessaires lors de la définition de modèles de contenus.

D'un point de vue typage de données, la spécification propose un ensemble important de types et une façon d'étendre ces types, pour une application donnée, à des choses plus complexes.

Finalement, un **Schema** est un modèle d'information défini sous forme électronique (comme l'était une **DTD**), mais avec des outils de modélisation plus puissants. Dans la sphère **XML**, le **Schema** pourra être utilisé comme outil de validation interactif de données, lors de la création de celles-ci, dans un éditeur, voire, plus tard, dans des **XForms**, sur Internet.

Des limites ? En lisant les listes de discussion, on peut trouver beaucoup de limitations exprimées sur ces possibilités de validation. Par exemple, d'aucuns parlent du fait de ne pas pouvoir valider un modèle de contenu en fonction d'une valeur d'attribut d'un élément englobant. C'est toute l'ambiguïté de l'utilisation des **Schema** qui est alors posée : est-ce un langage de programmation permettant d'exprimer des contraintes de validation ou est-ce un langage de définition de modèles ?

En avançant dans le sens de l'utilisation de modèles pour des activités d'ingénierie, et grâce aux nouvelles possibilités de typage, on peut avoir plusieurs **DTD** et **Schema**, fonctionnant sur une même classe de documents et, par des définitions de typage, déclencher dessus des processus applicatifs *ad hoc*. C'est ce qu'avaient fort bien compris les concepteurs de l'ICADD [sur internet : www.oasis-open.org/cover/gen-apps.html#icadd], pour rendre accessibles les documents aux non-voyants. Dans leurs spécifications [sur internet : www.oasis-open.org/]

cover/gen-apps.html#cadd], ils expliquent comment, en jouant seulement sur les attributs d'une DTD, on peut ajouter automatiquement de l'information aux documents, au moment de leur *parsing*, pour le plus grand profit des non-voyants qui pourront parcourir n'importe quel type de documents en se basant sur ces repères.

Informations connexes

Liste des types définis dans les schémas.


Catégorie	Type	Signification	Modèles de contenus	Attributs
Chaînes de caractères	string	Chaîne	O	O
	Name	Nom XML	O	O
	QName	Nom XML qualifié	O	O
	NCName	Nom XML non qualifié	O	O
Nombres	decimal	Décimal	O	O
	boolean	Valeur booléenne	O	O
	float	16bit floating point number	O	O
	double	32bit floating point number	O	O
	integer	Infinite accuracy integer	O	O
	nonPositiveInteger	Infinite accuracy integer less than 0	O	O
	negativeInteger	Infinite accuracy integer less than 0	O	O
	long	64bit integer	O	O
	int	32bit integer	O	O
	short	16bit integer	O	O
	byte	8bit integer	O	O
	nonNegativeInteger	Infinite accuracy integer more than 0	O	O
	positiveInteger	Infinite accuracy integer more than 1	O	O
	unsignedLong	Unsigned 64bit integer	O	O
unsignedInt	Unsigned 32bit integer	O	O	

Catégorie	Type	Signification	Modèles de contenus	Attributs
	unsignedShort	Unsigned 16bit integer	O	O
	unsignedByte	Unsigned 8bit integer	O	O
Temps	timeInstant	Date+Time	O	O
	timeDuration	Progress time	O	O
	recurringInstant		O	O
	date	Date	O	O
	time	Time	O	O
XML1.0 compatibilité	ID	XML1.0 ID	-	O
	IDREF	XML1.0 IDREF	-	O
	ENTITY	XML1.0 ENTITY	-	O
	NOTATION	XML1.0 NOTATION	-	O
	IDREFS	XML1.0 IDREFS	-	O
	ENTITIES	XML 1.0 ENTITIES	-	O
	NMTOKEN	XML 1.0 NMTOKEN	-	O
	NMTOKENS	XML 1.0 NMTOKENS	-	O
Autres.	binary	Binary	O	O
	uriReference	URI	O	O
	language	Language	O	O

Recommandations(s)


- ■ *XML Schema tome 0 : Introduction*
 Recommandation, version 20010502, du 02-05-2001
 Document sur <http://xmlfr.org/w3c/TR/xmlschema-0/>
- ■ *XML Schema tome 1 : Structures*
 Recommandation, version 20010502, du 02-05-2001
 Document sur <http://xmlfr.org/w3c/TR/xmlschema-1/>
- ■ *XML Schema Part 0: Primer*
 Recommandation, version Second Edition, du 28-10-2004
 Document sur <http://www.w3.org/TR/xmlschema-0/>
- ■ *XML Schema Part 1: Structures*
 Recommandation, version Second Edition, du 28-10-2004
 Document sur <http://www.w3.org/TR/xmlschema-1/>
- ■ *XML Schema Part 2: Datatypes*

Recommandation, version Second Edition, du 28-10-2004
Document sur <http://www.w3.org/TR/xmlschema-2/>

 *Associating Schemas with XML documents 1.0 (First Edition)*


Projet en cours, version 1.0, du 15-04-2010

Document sur <http://www.w3.org/TR/xml-model/>

 *Schema-Related Markup in Documents Being Validated*

Recommandation, version 20010502, du 02-05-2001

Document sur http://www.w3.org/TR/xmlschema-1/#Instance_Document_Constructions

 *XML Schema: Formal Description*


Projet en cours, version 20010320, du 20-03-2001

Document sur <http://www.w3.org/TR/xmlschema-formal/>

 *XML Schema: Component Designators*


Projet en cours, version 20050329, du 29-03-2005

Document sur <http://www.w3.org/TR/xmlschema-ref/>

 *Requirements for XML Schema 1.1*

Projet en cours, version 1.1, du 20030121

Document sur <http://www.w3.org/TR/2003/WD-xmlschema-11-req-20030121/>

 *Semantic Annotations for WSDL and XML Schema*

Recommandation, version 20070828, du 28-08-2007

Document sur <http://www.w3.org/TR/sawsdl/>