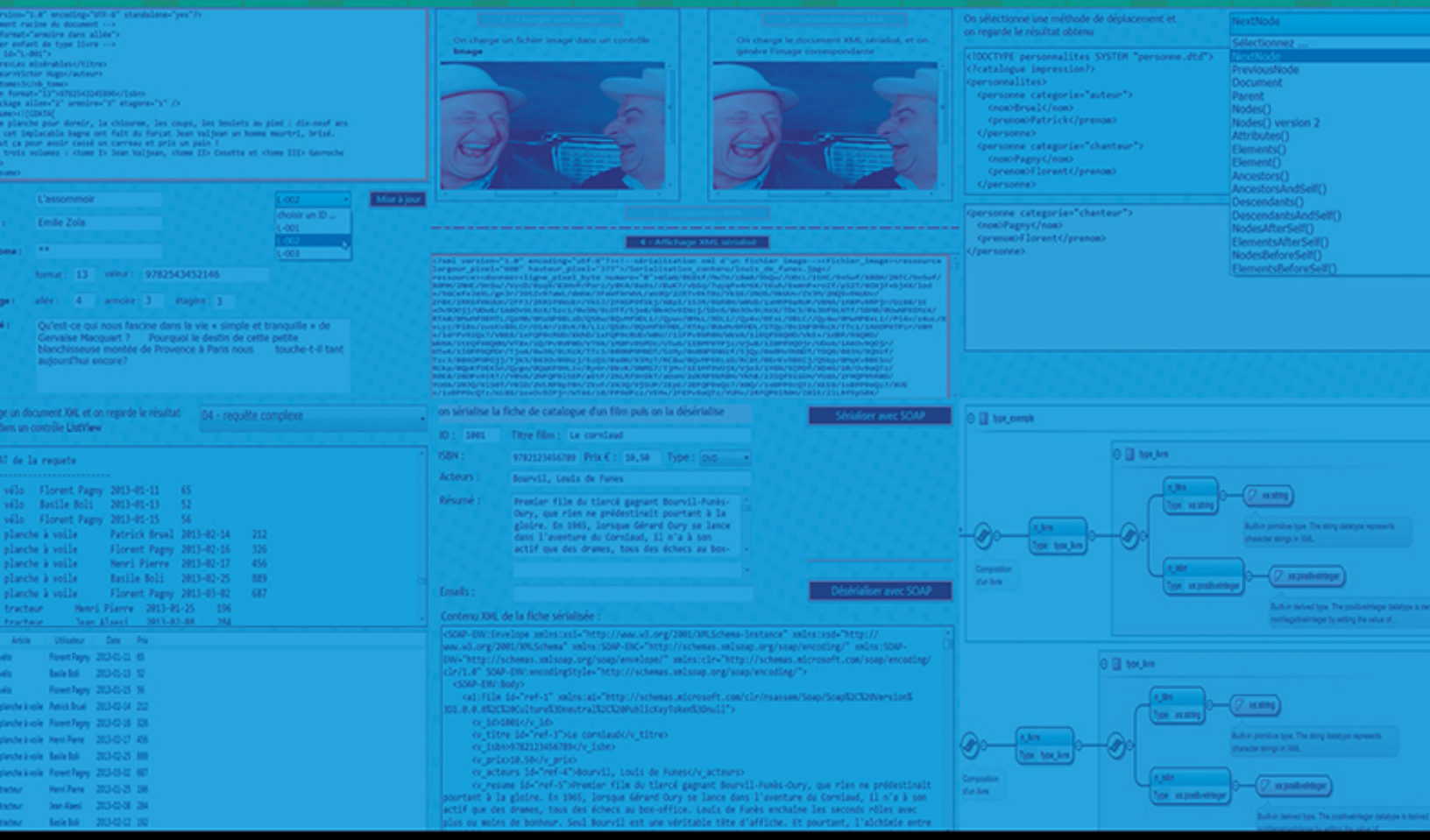
Programmation C#

XML

expliqué et illustré

avec

C#5, WPF et LINQ



Avec l'environnement de développement intégré
Visual Studio 2012 de MICROSOFT

Patrice REY

Table des matières

Le code source de programmation est téléchargeable gratuitement à l'adresse internet http://www.reypatrice.fr

Avant-propos

Cet ouvrage s’inspire de mon expérience issue de plusieurs années d’enseignement et de formation en informatique. J’ai souhaité faire bénéficier de cette expérience tous ceux qui, à des titres divers, peuvent être amenés à étudier XML, ou à mener à bien des développements qui impliquent son utilisation.

XML est une famille de langages partageant des caractéristiques communes, et dédiés à une multitude d’usages divers. Les facilités d’écriture de ce format, les possibilités de traitement des données qu’il offre et sa souplesse d’utilisation, en font un format extrêmement bien adapté aux échanges de données entre applications aussi bien qu’à leur simple stockage. C’est donc tout naturellement que son usage s’est largement répandu au point d’être aujourd’hui incontournable.

Cet ouvrage propose donc d’en aborder l’apprentissage de manière progressive et pédagogique, et de pouvoir en apprécier son utilisation en programmation, notamment avec C#5, WPF et LINQ.

Le contenu du livre

Les différents chapitres permettent d’apprendre à concevoir des documents XML, à valider ces documents XML, à générer des documents XML au travers des langages de transformation et en fonction d’expression XPath, et à

utiliser en pratique ces documents XML dans des applications écrites en C#5 à l'aide de WPF et LINQ.

Les chapitres de la première partie abordent les points suivants:

- le chapitre 1 expose les bases du langage XML par la structuration de l'information, les règles d'écriture, les notions de document bien formé et valide, et la mise en forme simple à l'aide du langage CSS.
- le chapitre 2 traite du langage XPath pour l'élaboration et l'évaluation d'expression sur un document XML, au travers de sa syntaxe, de ses opérateurs et de ses fonctions.
- le chapitre 3 traite de la réalisation des feuilles de transformations XSLT nécessaires à l'élaboration de fichier HTML pour visualiser des données XML; il sera abordé l'utilisation des modèles, l'utilisation des instructions courantes et avancées, la création des éléments, les tests conditionnels et les tris alphanumériques.
- le chapitre 4 traite de la réalisation des feuilles de transformation XSL-FO, et de leurs usages dans la réalisation de PDF imprimables avec Apache FOP; un cas pratique complet de toute la chaîne de transformation sera abordé avec l'utilisation d'un logiciel commercial qu'est OXYGEN Editor 15.
- le chapitre 5 traite de la conception et de la réalisation des DTD (définition de type de document); les DTD évitent les dysfonctionnements dans la conception des documents XML; nous y verrons comment associer une DTD à un document XML, quelle est la syntaxe de ses règles, et comment valider un document XML.
- le chapitre 6 traite d'un autre langage de définition de contenu qu'est le *XML Schema* (le schéma XML en français); le XML Schema est de loin le plus utilisé; nous

y verrons comment déclarer des éléments et des attributs, quels sont les types simples définis dans la recommandation du W3C et ceux définis par dérivation, comment concevoir les types complexes de données, et comment modulariser et documenter le schéma XML.

Les chapitres de la deuxième partie abordent les points suivants:

- le chapitre 7 traite des manipulations à connaître pour un document XML avec le modèle objet de document (DOM); on y verra notamment comment charger un document XML pour visualiser son contenu, comment effectuer des recherches de noeuds, comment effectuer des modifications de noeuds, et comment visualiser l'arborescence d'un document XML dans un contrôle dédié *TreeView*.
- le chapitre 8 traite des manipulations de lecture et d'écrire avec les classes *XmlTextReader* et *XmlTextWriter*; il sera abordé les déplacements dans l'arborescence XML avec *XmlTextReader*, et l'écriture des données XML avec *XmlTextWriter*.
- le chapitre 9 traite de la mise en pratique de l'évaluation des expressions XPath avec notamment comment élaborer une requête pour évaluer une expression XPath, comment lire les données retournées par la requête, et comment les modifier et les sauvegarder.
- le chapitre 10 traite de la pratique de la validation de document XML par la DTD et par le schéma XML; il sera vu notamment comment générer un schéma XML par programmation grâce à des classes spécifiques du *framework* .NET.
- le chapitre 11 traite de la pratique de la transformation XSLT par programmation; il sera vu notamment comment effectuer une transformation XSLT par

programmation pour obtenir un document HTML, ainsi que le cas avec le passage de paramètres.

- le chapitre 12 traite de l'utilisation de l'API LINQ To XML; après avoir pris connaissance du modèle objet de LINQ To XML, vous apprendrez la création complète d'arbres XML grâce aux différents éléments qui composent LINQ To XML; vous verrez comment lire et sauvegarder des fichiers XML, comment se déplacer dans l'arborescence XML, comment modifier des données XML, et comment gérer les annotations et les événements sur les éléments XML.
- le chapitre 13 traite des opérateurs de requête spécifiques de LINQ To XML; après avoir pris connaissance de la classe *Extensions*, nous verrons l'utilisation pratique de l'ensemble de ces opérateurs spécifiques de LINQ To XML.
- le chapitre 14 traite des utilisations courantes pratiquées avec LINQ To XML; il sera notamment vu comment visualiser le résultat de requête dans un contrôle *ListView*, comment utiliser les clauses dans la rédaction des requêtes, comment simplifier les tâches de conception d'arbres XML avec la construction fonctionnelle, et comment effectuer la validation de document XML avec les schémas XML.
- le chapitre 15 traite de la sérialisation XML et de la sérialisation SOAP; vous verrez notamment comment est réalisé le processus de sérialisation et le processus de désérialisation aux formats XML et SOAP, avec la possibilité de personnaliser les balises et d'encoder les chaînes en *base64*.

Toutes les notions abordées dans ce livre font l'objet d'une utilisation directe et pratique que vous trouverez dans le code source en téléchargement. Il est ainsi possible de suivre aisément la réalisation technique des composants en fonction des chapitres du livre. C'est une démarche

volontaire, dans un but pédagogique, pour progresser rapidement.

Les logiciels requis pour le développement

Pour réaliser plus facilement des programmes en C# et XAML, il est préférable d'utiliser un environnement de développement intégré ou IDE (*Integrated Development Environment*). Cet IDE permet de développer, de compiler et d'exécuter un programme dans un langage donné qui sera, dans ce livre, le C# (se prononce C Sharp) dans sa version 5.

Microsoft propose comme IDE soit Visual Studio 2012 dans sa version payante avec 90 jours d'essai gratuit (figure A1), soit Visual Studio 2012 Express dans sa version allégée mais gratuite (figure A2).

Puissante interface IDE garantissant la production d'un code de qualité, Visual Studio 2012 intègre de nouvelles fonctionnalités qui simplifient le processus de développement des applications, de la conception au déploiement.

Pour télécharger et installer l'environnement de développement intégré, ouvrez votre navigateur et allez sur la page web suivante: *http://www.microsoft.com/france/visual-studio/essayez/express.aspx* pour la version gratuite, ou bien sur *http://www.microsoft.com/france/visual-studio/essayez/download.aspx* pour la version d'essai complète.

FIGURE A1

TÉLÉCHARGEMENTS DE VISUAL STUDIO 2012

Télécharger une version d'évaluation complète.

Essayer Visual Studio 2012 Ultimate

Essayer Visual Studio 2012 Professional

Tester Visual Studio 2012

Découvrez ce puissant outil et ses fonctionnalités en téléchargeant une version d'essai complète de Visual Studio 2012.

Version d'évaluation de Visual Studio 2012 Ultimate

Découvrez comment Visual Studio 2012 Ultimate peut vous aider à architecturer, créer, tester, déboguer et gérer des solutions logicielles. En savoir plus sur Visual Studio 2012 Ultimate.

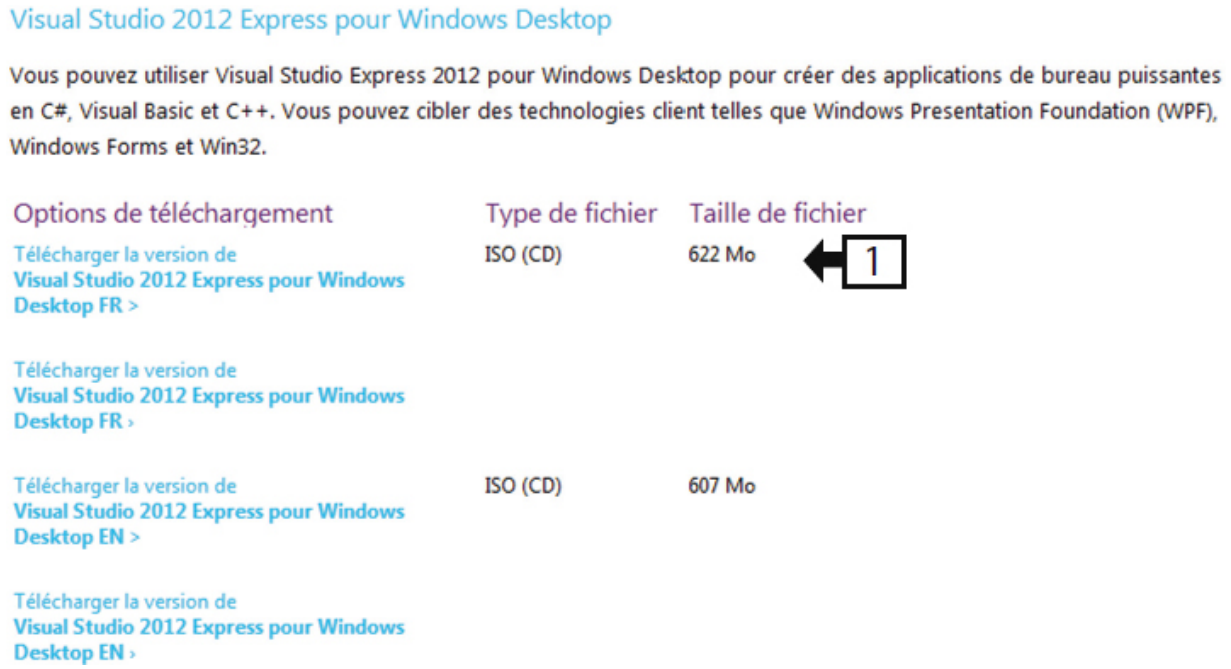
Options de téléchargement	Type de fichier	Taille de fichier
Télécharger la version d'évaluation de **Visual Studio 2012 Ultimate FR >**	ISO (DVD-5)	1,5 Go
Télécharger la version d'évaluation de **Visual Studio 2012 Ultimate FR >**		
Télécharger la version d'évaluation de **Visual Studio 2012 Ultimate EN >**	ISO (DVD-5)	1,5 Go
Télécharger la version d'évaluation de		

1

Prêt à acheter ?

Nos revendeurs peuvent vous conseiller sur les meilleurs termes.

FIGURE A2



Les liens de téléchargement

Tout le code source de programmation concernant cet ouvrage pourra être téléchargé gratuitement à l'adresse web suivante, en allant sur la fiche du livre:

http://www.reypatrice.fr

Bonne lecture à tous.

Partie 1

Tout savoir sur le langage XML

1

Les principes de XML

Au sommaire de ce chapitre

- XML, qui est dérivé de SGML, est l'héritier attendu de SGML et de HTML.
- Installer un éditeur de XML.
- Concevoir un document XML avec le prologue, les commentaires, l'arborescence du document, les noeuds de type élément et de type attribut.
- Utiliser les entités prédéfinies et les sections CDATA et PCDATA.
- Comprendre l'encodage de document.
- Bien formé un document XML, et l'afficher dans un navigateur web par défaut et avec une mise en forme CSS.

XML est un langage de description de document. C'est un langage qui va contenir à la fois les données du contenu mais aussi une qualification représentée par les métadonnées sur ce contenu. C'est un langage qui va donc décrire un document ou des données en mélangeant les valeurs et la qualification de ces valeurs.

XML contient beaucoup de richesses. Nous verrons comment structurer un document XML, quelles sont les contraintes qui font qu'un document XML est bien formé et reconnaissable comme un vrai document XML, qu'est-ce que l'on met à l'intérieur d'un document XML, quels sont les

types de données, comment est-ce que l'on va représenter les données plus complexes et les caractères spéciaux, etc.

Nous verrons comment manipuler le document XML car autour de XML, beaucoup de technologies se sont développées, technologies qui permettent de qualifier les métadonnées (avec les espaces de noms par exemple), technologies qui permettent de valider des structures par rapport à des règles spécifiques (d'une entreprise par exemple) pour des types de données spécifiques à l'aide de ce que l'on appelle un schéma.

Nous verrons aussi que l'on peut parcourir le chemin à l'intérieur de XML pour aller en extraire des informations directement avec un langage qui s'appelle XPATH. Nous verrons qu'à l'aide de ce chemin, nous pouvons extraire des informations mais également appliquer des feuilles de style de façon à faire des transformations pour extraire un fragment et l'afficher dans un autre format comme du HTML par exemple.

1 - La genèse de XML

La question que tout le monde se pose est bien entendu qu'est-ce que XML ? Est-ce que XML est un langage ? Oui c'est un langage mais pas au sens où on l'entend en terme de langage de programmation. Ce n'est pas un langage qui fait quelque chose, qui effectue des actions, c'est un langage qui décrit un document. Il s'agit en fait d'un langage de balisage de document, dérivé d'un langage préliminaire qui s'appelle le SGML.

A la fin des années 1970, la société IBM a été à l'origine de la création d'un langage de description de données, fondé sur des balises. Ce langage nommé SGML (*Standard Generalized Markup Language*) fut publié comme norme *ISO 8879:1986* en 1986. Ce langage permettait de décrire la structure d'un document indépendamment de sa

visualisation, ou plus généralement de son interprétation par une application tierce. Il fallait en effet fournir:

- une *définition de type de document* (DTD) afin de définir les éléments autorisés dans le document à décrire.
- une instance du document c'est-à-dire le texte réel du document comprenant les éléments SGML définis dans la DTD et qui identifient les diverses parties du texte; même si une instance de document peut partager une DTD avec d'autres documents, elle ne peut se conformer qu'à une seule DTD.
- la synthèse du document qui sert à préciser les principaux aspects de l'application SGML; c'est à ce niveau que sont déterminées les options et que sont précisés le jeu de caractères utilisé ainsi que les autres fonctions similaires.

SGML est un puissant langage de description mais il est très complexe, notamment à cause des règles concernant les fermetures codées, les fermetures implicites, etc. Lors des travaux de développement du *World Wide Web*, il fut nécessaire de simplifier le SGML pour le rendre utilisable facilement pour la création de pages web. Il en résulta HTML, une application de SGML limitée à un contexte particulier. Il faudra attendre la version HTML 4.0 publiée par le consortium W3C, en charge de la définition des standards de l'Internet, en 1997 pour avoir un format HTML dans lequel on ait la séparation du fond et de la forme par l'emploi des *feuilles de style* à l'aide du langage CSS.

Le HTML est une forme de SGML qui décrit le formatage d'un document. L'interprétation des balises HTML va permettre de réaliser des tableaux, des listes, des paragraphes, etc. Le navigateur va lire la source HTML et va produire un document formaté en fonction des balises trouvées et interprétées. Le navigateur apparait comme un

simple outil consistant à interpréter le langage HTML et à le reformater pour l'utilisateur sur son ordinateur client.

Comme HTML restait évidemment un langage étroitement lié à la publication de contenu pour le web et non personnalisable, il fut décidé de tirer parti à la fois de la souplesse de SGML et de la simplicité d'utilisation de HTML. Ce fut chose faite avec la publication en février 1998 de la première version de la recommandation XML 1.0.

XML, qui est dérivé de SGML, est l'héritier attendu de SGML et de HTML. XML est une simplification de SGML avec un certain nombre de concepts SGML conservés et parfois quelques aménagements. Avec XML, il est maintenant facultatif de fournir une DTD tout en gardant la possibilité d'en créer une, voire d'utiliser un autre format comme *XML Schema* pour définir les éléments utilisables dans le document XML. La séparation du fond et de la forme est stricte, rendant possible le traitement d'un même document XML par diverses applications en vue d'utilisations différentes. En revanche, XML présente quelques nouveautés:

- une syntaxe plus stricte qu'en SGML, ce qui réduit les contraintes sur les applications de traitement.
- une gestion de l'encodage qui autorise l'utilisation de caractères spéciaux.
- un langage de transformation XSL (*eXtensible Stylesheet Language*), qui est lui-même un format XML, qui permet d'automatiser le traitement d'un document XML donné.
- un langage de description et de définition, *XML Schema*, qui est lui aussi un document XML, qui étend les possibilités des DTD.

XML va décrire non pas le formatage mais le type de contenu qui va être proposé dans le document. Il s'agit donc d'une *description sémantique* du contenu. XML va servir à

qualifier les différentes données, les différents éléments qui sont dans un document. C'est donc très intéressant car cela va permettre de pouvoir échanger un document non seulement avec son contenu mais aussi avec la *description intégrée* de son contenu. XML permet de séparer nettement le fond de la forme d'un document. Il s'avère souvent nécessaire de transformer le fichier brut par exemple pour permettre son affichage ou son impression pour l'oeil humain, ou bien son traitement par une application donnée. Cela entraîne la nécessité de faire intervenir au moins deux fichiers: le fichier XML proprement dit qui contiendra les données à mettre en forme; et le fichier qui permettra sa transformation et son adaptation au média de sortie sélectionné.

Dans le contexte d'une production de masse, l'ensemble de la chaîne est constitué de trois à quatre maillons:

- la définition des éléments utilisables dans le document XML.
- le document XML lui-même.
- les outils de transformation du document.
- dans le contexte d'une page web, une feuille de style CSS pour la mise en forme finale.

2 - Pourquoi XML ?

La question que l'on peut se poser c'est pourquoi utiliser XML, à quoi cela peut-il bien servir ? XML a un certain nombre d'avantages. Vous pouvez construire des documents, des choses statiques en pur texte, qui contiennent non seulement un contenu qui peut être relativement riche, long et important, mais également la description de ce contenu. On peut donc ajouter la sémantique à l'intérieur de ce document. Cela peut aboutir à des choses intéressantes comme ce que l'on appelle le *web sémantique*, c'est-à-dire la capacité à travers des

pages web non seulement de formater mais aussi de qualifier la donnée. Cela permet à des moteurs de recherche automatisés de mieux cibler le contenu des pages web, et par conséquent de faire des recherches plus précises. En mélangeant le formatage et la sémantique, on enrichit la donnée qui se trouve sur Internet.

Il s'agit de donner du sens à des documents ou à de la donnée, et il s'agit aussi de pouvoir se les échanger. L'avantage de XML c'est qu'il fonctionne avec du texte. Le texte en informatique est reconnu par tous les systèmes, par tous les langages. Si j'échange un document qui est purement en texte, je vais être beaucoup plus portable, beaucoup plus universel que si j'échange un document de type binaire comme un fichier Word ou un fichier PDF. Le texte est alors idéal. Je vais pouvoir échanger en pur texte des données qui sont qualifiées et qui sont aussi hiérarchiques parce qu'un élément que l'on appelle un noeud, va pouvoir contenir des sous-éléments (des sous-noeuds). La hiérarchie permet d'exprimer quelque chose de complexe avec des données structurées.

Avec XML, on est plus dans une approche où l'on va construire des documents qui sont plutôt des données que l'on va manipuler par des programmes, et que l'on va échanger. On va pouvoir exprimer des données riches, qui vont être bien structurées ou semi-structurées, et les traiter automatiquement par un certain nombre de systèmes car le langage XML va être très facile à parser. Parser un document XML (*parsing* en anglais) consiste à interpréter automatiquement par des bibliothèques le contenu XML. Ces bibliothèques d'interprétation sont présentes dans tous les langages de programmation.

De plus, il y a beaucoup de techniques qui tournent autour de XML, notamment des techniques qui vont permettre, outre le fait de parser, de reconnaître les différents éléments mais aussi de les transformer, de prendre un document et d'utiliser un langage de

description, de transformation qui va permettre d'élaborer automatiquement un nouveau document à partir du document originel (il s'agit de XSLT pour le langage de transformation de XML).

On va pouvoir également faire des recherches automatiquement à l'intérieur du document ou accéder à certains noeuds de ce document, à l'aide d'un langage qui s'appelle XPATH, qui permet de se déplacer, de trouver des chemins à l'intérieur du document. Et une extension encore plus puissante, qui s'appelle XQUERY, permet de faire des recherches à l'intérieur du document XML.

On va pouvoir indiquer aussi qu'un document a une certaine structure obligatoire à l'aide d'un langage de description de structure du document qui s'appelle le schéma XML (*XML Schema*). Si j'échange des données entre différents systèmes à l'aide du schéma, je vais pouvoir indiquer que le document XML doit contenir un certain nombre de balises, avoir une certaine structure bien définie de façon à s'assurer que le document est valide du point de vue de cette structure.

Comme vous le voyez, nous avons non seulement un langage de description du document, mais aussi des outils qui s'ajoutent au XML pour en assurer les transformations, les recherches et la validation. Il s'agit d'un ensemble bien construit d'outils qui vont nous permettre de faire des échanges de données solides, multiplateformes, très souples et extensibles à l'infini.

3 - Installer un éditeur de XML

Ecrire du code XML peut se faire avec un simple éditeur de texte. Mais, comme le XML possède une syntaxe stricte de balisage, il est alors plus facile d'utiliser des éditeurs dédiés de XML, permettant de réduire considérablement le code à écrire. Par exemple, lorsque vous écrivez une balise

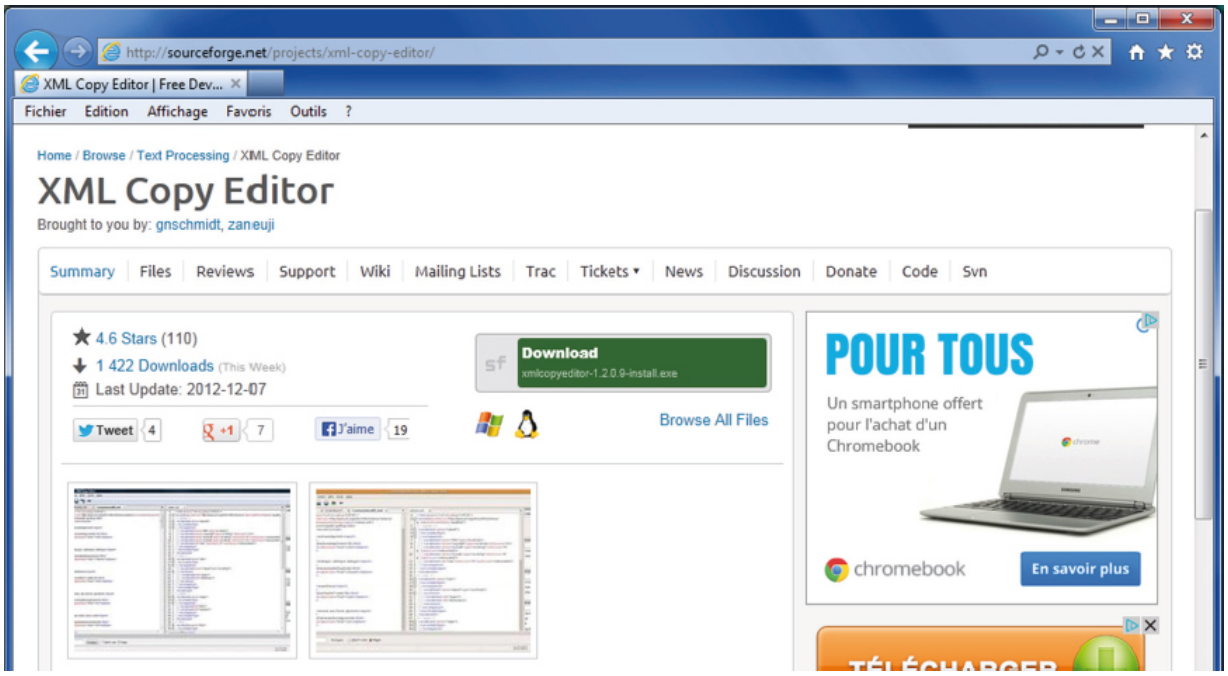
ouvrante, l'éditeur complète automatiquement en écrivant la balise fermante. C'est une aide très utile à l'écriture.

De nombreux logiciels commerciaux existent, ils sont en général très complet. Le plus connu et le plus complet est *OXYGEN XML EDITOR*. Il existe aussi des logiciels plus simples et *open source*, très facile à l'utilisation notamment pour l'apprentissage du XML. Ici nous utiliserons le logiciel *XML Copy Editor*, qui est un logiciel *open source* très simple et très pratique dans le cadre de l'apprentissage de XML, doté d'une translation linguistique en français. Vous pouvez le télécharger en vous rendant sur la page web suivante (figure 1.1):

http://sourceforge.net/projects/xml-copy-editor/

L'installation du logiciel s'effectue en quelques minutes en utilisant les réglages proposés par défaut.

FIGURE 1.1



4 - La conception d’un document XML

Supposons que nous ayons une bibliothèque de livres (figure 1.2) composée de 4 allées, et chacune de ces 4 allées contient un ensemble de 4 armoires à livres. Une armoire à livres contient un ensemble d’étagères (par exemple une dizaine).

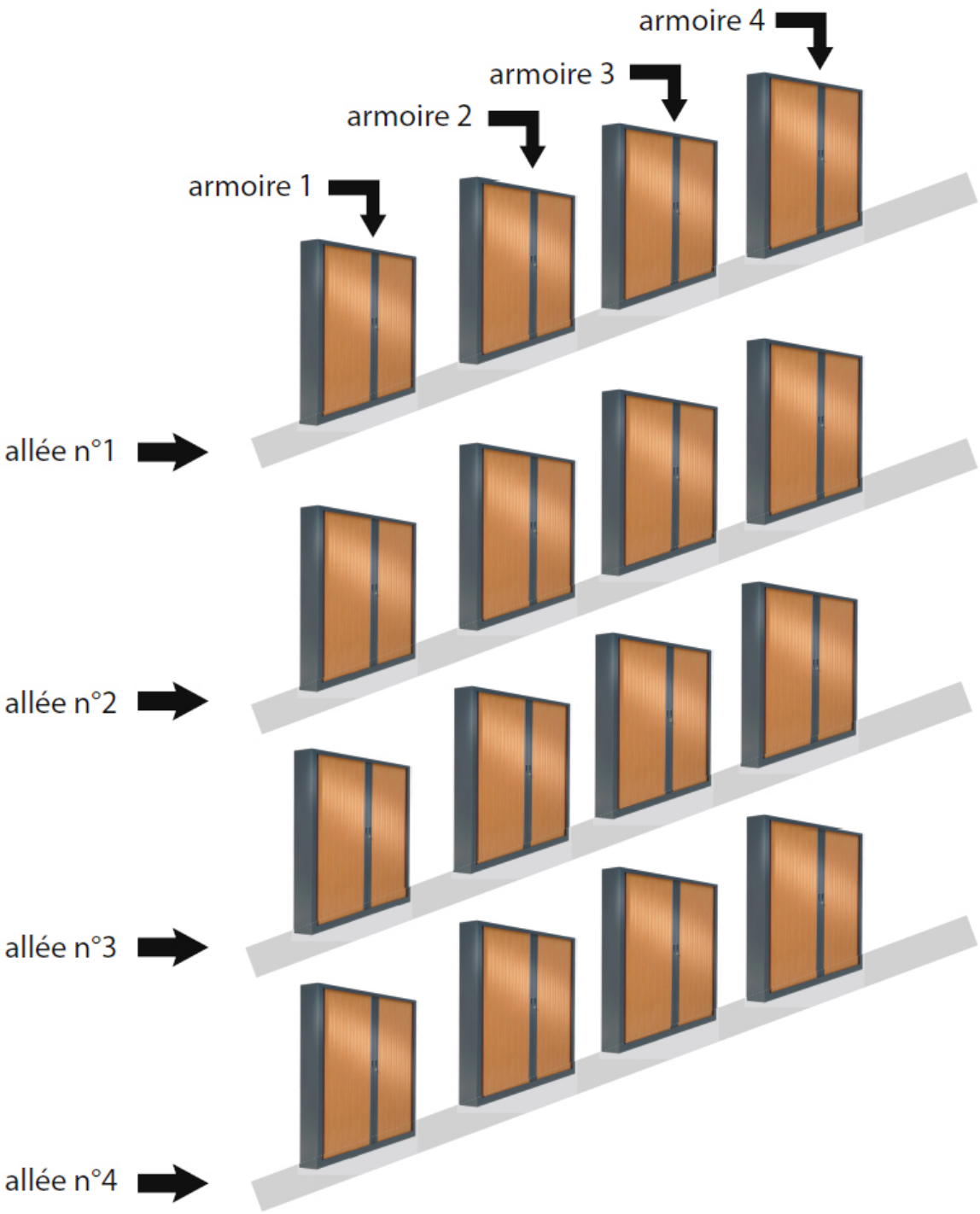


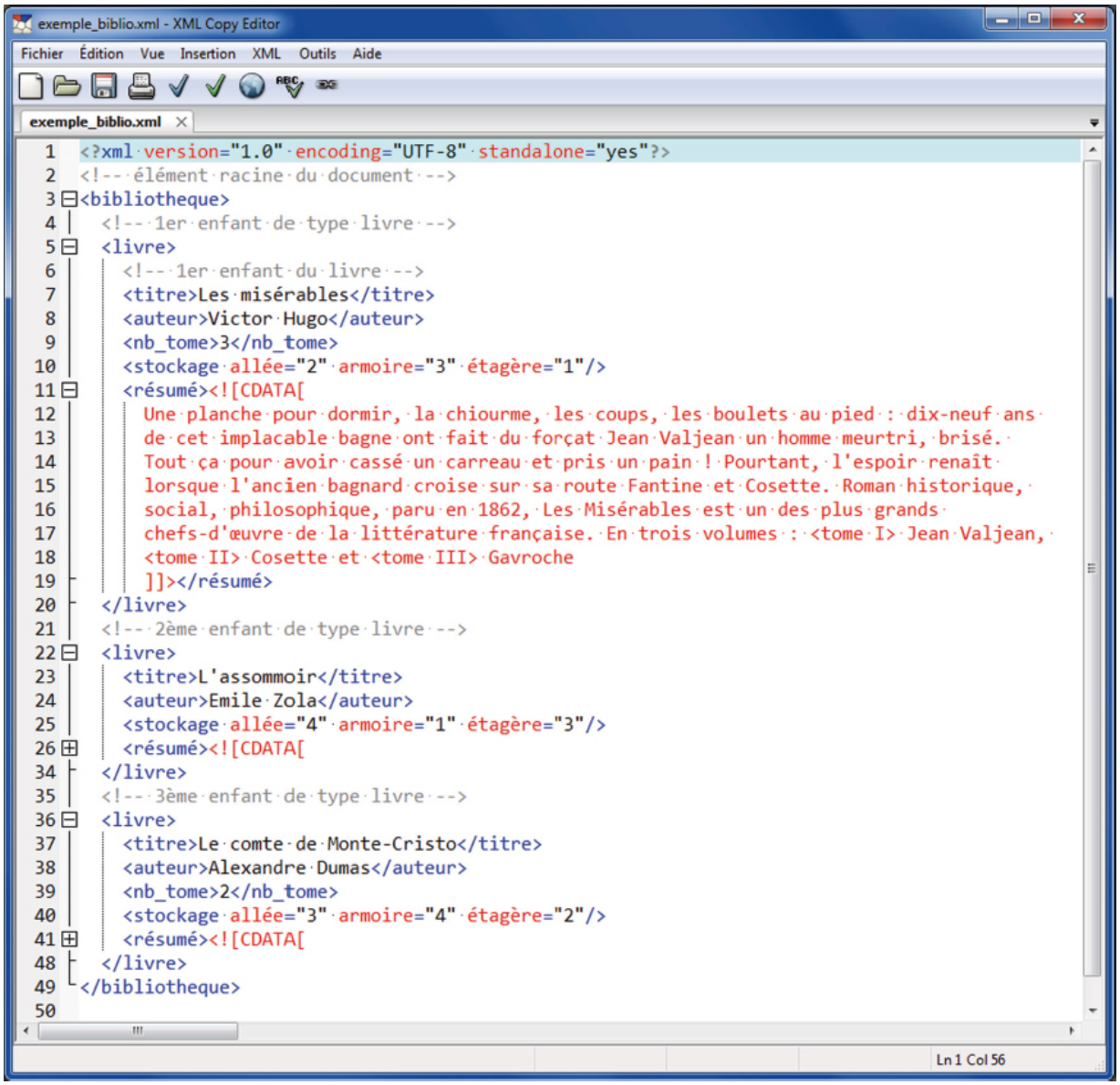
FIGURE 1.2

Nous allons créer un document XML pour exprimer notre bibliothèque avec son contenu. Le but est d'avoir une énumération des livres et de pouvoir les localiser en fonction des allées, des armoires et des étagères. La figure

1.3 visualise un exemple de document XML tel que l'on pourrait le réaliser avec *XML Copy Editor*.

Ce document XML, intitulé *exemple_biblio.xml*, se trouve dans le dossier *chapitre_01* du code source de programmation. Vous pouvez télécharger gratuitement ce code source en vous rendant sur la page web dédiée à ce livre (la fiche du livre se trouve sur le site web à l'adresse *http://www.reypatrice.fr*).

FIGURE 1.3

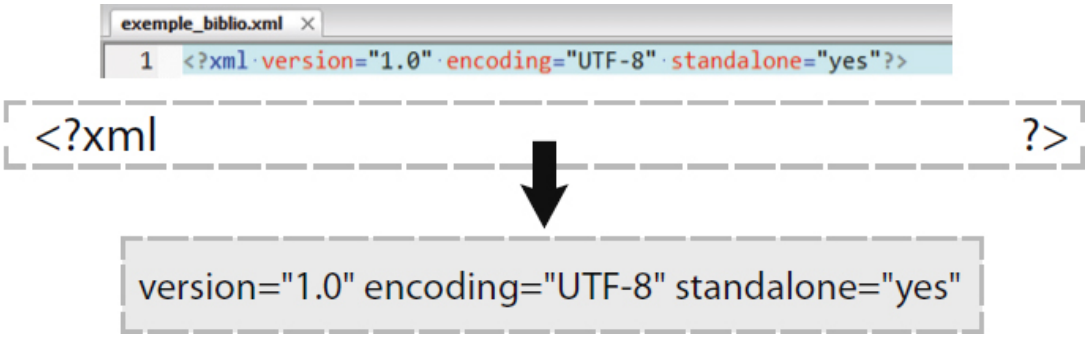


Le document *exemple_biblio.xml* est un fichier dont les données sont structurées à l'aide de «briques» que nous allons passer en revue.

4.1 - Le prologue

Notre document XML commence par la ligne *<?xml version= «1.0» encoding= «UTF-8» standalone= «yes»?>*. Cette ligne s'appelle le *prologue* du document XML (figure 1.4). Le prologue d'un document XML se place toujours en première ligne du fichier. A noter que l'écriture du prologue est facultative.

FIGURE 1.4



Le but du prologue est de préciser le numéro de version du format XML contenu, de préciser l'encodage de caractères utilisé, et de préciser si le document XML dépend d'un autre fichier pour la définition de sa syntaxe. Le prologue est ce que l'on appelle une *instruction de traitement* particulière.

Le prologue est représenté par une balise spéciale qui commence par *<?xml* et se finit par *?>*. Cette balise contient des attributs qui sont *version*, *encoding* et *standalone*. Un attribut commence par une chaîne de caractères (c'est le nom de l'attribut), et il possède une valeur indiquée entre guillemets. Ici par exemple l'attribut *version* possède la valeur *1.0*. Le numéro de version *1.0* du format XML est celui qui sera toujours utilisé (attribut *version = «1.0»*).

L’attribut *encoding* précise l’encodage de caractères utilisé pour réaliser le document XML. Il s’agit d’une information importante qui est transmise à l’application chargée du traitement, et qui lui indique dans quel type de codage le fichier a été enregistré. Autrement dit, cet attribut *encoding* indique comment physiquement sur le disque dur une suite de *bits* est associée à un caractère ou un autre. L’encodage utilisé par défaut est l’UTF-8. Rien n’empêche d’utiliser un autre encodage (comme ISO-8859-1, ASCII, etc.) du moment que cela est précisé dans l’attribut *encoding* et que l’enregistrement du fichier est bien effectué avec cet encodage spécifié.

Une dernière information est précisée par l’attribut *standalone*. Cet attribut *standalone* est facultatif dans l’écriture du prologue. Sa valeur par défaut est *yes*. Il indique si le document XML dépend d’un autre fichier pour la définition de sa syntaxe, ou bien s’il se suffit à lui-même. La valeur *yes* par défaut indique que le document tient «debout tout seul», autrement dit qu’il n’est pas nécessaire de lui adjoindre un autre document pour vérifier sa syntaxe. La seconde valeur possible pour cet attribut est *no*, signifiant que le document a besoin d’un autre fichier sous forme d’une DTD (nous verrons dans un autre chapitre comment s’articule les définitions de type de document, les DTD). A noter une chose importante: dans un prologue, les attributs doivent être impérativement utilisés dans l’ordre *version*, *encoding* et *standalone*.

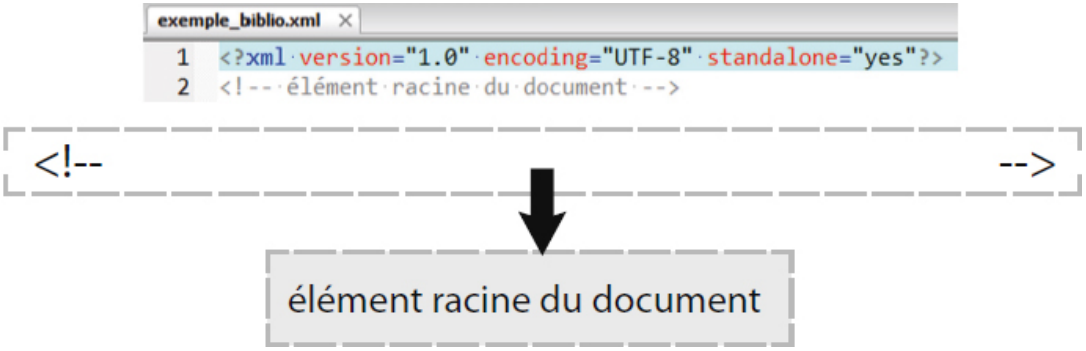
4.2 - Les commentaires

Les commentaires sont conformes à la norme SGML. Un commentaire commence par la chaîne de caractères *<!--* et se termine par la chaîne *-->* (figure 1.5). Du fait de la composition du balisage d’un commentaire, un commentaire ne peut pas contenir la chaîne «--» car c’est une chaîne réservée.

Un noeud commentaire sert bien évidemment à ajouter un commentaire au document XML. Les commentaires n’ont pas vocation à être interprétés par l’application chargée du traitement du document. Ils peuvent par conséquent servir à insérer une documentation succincte dans le corps du document XML.

A noter que l’on retrouve cette syntaxe de commentaires dans un document HTML. Ils sont ignorés lors de leur affichage par le navigateur.

FIGURE 1.5



4.3 - L'arborescence du document

Lahiérarchisationdesdonnéesaune traductionphysique danslastructuredudocument XML lui-même. On parle de *l'arborescence du document* en faisant référence à cette structure pyramidale. Cette arborescence est constituée de noeuds. L'arborescence du document représente donc la structure hiérarchique des noeuds.

La plupart des noeuds d'un document ont un noeud parent, parfois des noeuds frères, parfois aussi des noeuds enfants. Un unique noeud de l'arborescence ne possède pas de parent, c'est l'élément *racine*. De lui seul dépendent tous les autres noeuds à des niveaux divers et variés de l'arborescence.

La figure 1.6 schématise un exemple d'arborescence de document XML. Le noeud racine possède 3 enfants qui sont le noeud 1, le noeud 2 et le noeud 3. Le noeud 2 possède 2 enfants (le noeud 2.1 et le noeud 2.2), et le noeud 3 possède 3 enfants (le noeud 3.1, le noeud 3.2 et le noeud 3.3). Les noeuds 1, 2 et 3 sont des noeuds frères, et ils possèdent le même parent qui est le noeud racine. Les noeuds 2.1 et 2.2 sont des noeuds frères et ils possèdent comme noeud parent le noeud 2.

FIGURE 1.6