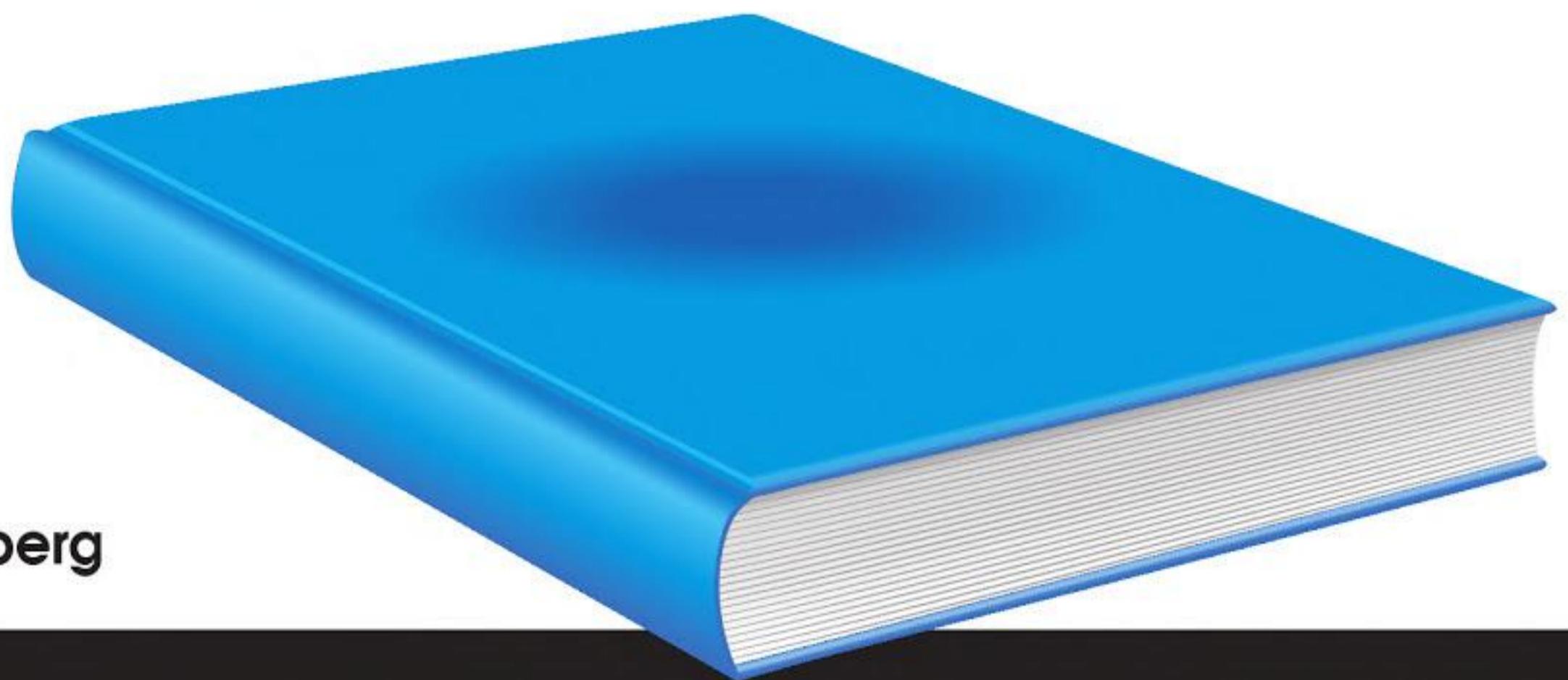


Le Programmeur

Manuel  
de prise en  
main de **XML**



Kevin Howard Goldberg

Ouvrage en couleurs !

PEARSON

Auteursrechtelijk beschermd materiaal

---

Pearson Education France a apporté le plus grand soin à la réalisation de ce livre afin de vous fournir une information complète et fiable. Cependant, Pearson Education France n'assume de responsabilités, ni pour son utilisation, ni pour les contrefaçons de brevets ou atteintes aux droits de tierces personnes qui pourraient résulter de cette utilisation.

Les exemples ou les programmes présents dans cet ouvrage sont fournis pour illustrer les descriptions théoriques. Ils ne sont en aucun cas destinés à une utilisation commerciale ou professionnelle.

Pearson Education France ne pourra en aucun cas être tenu pour responsable des préjudices ou dommages de quelque nature que ce soit pouvant résulter de l'utilisation de ces exemples ou programmes.

Tous les noms de produits ou marques cités dans ce livre sont des marques déposées par leurs propriétaires respectifs.

Publié par Pearson Education France  
47 bis, rue des Vinaigriers  
75010 PARIS  
Tél. : 01 72 74 90 00  
[www.pearson.fr](http://www.pearson.fr)

Réalisation p.a.o. : Léa B.  
ISBN : 978-2-7440-2369-9  
Copyright © 2009 Pearson Education France  
Tous droits réservés

Titre original :  
*Visual QuickStart Guide XML, second edition*  
Traduction : Eric Jacoboni  
Avec la contribution technique de : Gilles Hunault

ISBN original : 978-0-321-55967-8  
Copyright © 2009 by Elizabeth Castro  
and Kevin Howard Goldberg

All rights reserved

Édition originale publiée par Peachpit Press  
1249 Eighth Street  
Berkeley, CA 94710  
[www.peachpit.com](http://www.peachpit.com)

Aucune représentation ou reproduction, même partielle, autre que celles prévues à l'article L. 122-5 2° et 3° a) du code de la propriété intellectuelle ne peut être faite sans l'autorisation expresse de Pearson Education France ou, le cas échéant, sans le respect des modalités prévues à l'article L. 122-10 dudit code.

No part of this book may be reproduced or transmitted in any form, by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the publisher, except for the inclusion of brief quotations in a review.

# Table des matières

<b>Introduction</b> .....	IX
Qu'est-ce que XML ? .....	X
<u>Puissance de XML</u> .....	XI
Extension de XML .....	XII
XML en pratique .....	XIII
À propos de ce livre .....	XIV
Ce que vous ne trouverez pas dans ce livre .....	XVI

## PARTIE 1 - XML

<b>CHAPITRE 1. Créer des documents XML</b> ....	3
Exemple de document XML .....	4
Règles d'écriture des documents XML .....	5
Éléments, attributs et valeurs .....	6
<u>Commencer un document XML</u> .....	7
L'élément racine .....	8
Les éléments fils .....	9
<u>Attributs</u> .....	11
Éléments vides .....	12
Commentaires .....	13
Entités prédefinies – cinq symboles spéciaux ...	14
<u>Affichage des éléments sous forme de texte</u> ....	15

## PARTIE 2 - XSL

<b>CHAPITRE 2. XSLT</b> .....	19
Transformer XML avec XSLT .....	20
Création d'une feuille de style XSLT .....	22
<u>Création de la règle racine</u> .....	23
Production de HTML .....	24
Affichage de valeurs .....	26
Boucles sur les nœuds .....	28

Traitement conditionnel des nœuds .....	30
<u>Utilisation de plusieurs choix</u> .....	31
Tri des nœuds avant traitement .....	32
Production d'attributs .....	33
Création et application de règles .....	34

<b>CHAPITRE 3. Motifs et expressions XPath</b> ...	37
Localisation des nœuds .....	38
Déterminer le nœud courant .....	40
Désigner le nœud courant .....	41
Sélectionner le fils d'un nœud .....	42
<u>Sélectionner le père ou le frère d'un nœud</u> .....	43
Sélectionner les attributs d'un nœud .....	44
Choisir des nœuds .....	45
Créer des chemins d'accès absolus .....	46
<u>Sélectionner tous les descendants</u> .....	47

<b>CHAPITRE 4. Fonctions XPath</b> .....	49
Comparer deux valeurs .....	50
<u>Tester la position</u> .....	51
Multiplier, diviser, ajouter et soustraire .....	52
Compter les nœuds .....	53
Formater les nombres .....	54
<u>Arrondir les nombres</u> .....	55
Extraire des sous-chaînes .....	56
Modifier la casse d'une chaîne .....	57
Additionner des valeurs .....	58
Autres fonctions XPath .....	58

<b>CHAPITRE 5. XSL-FO</b> .....	61
Les deux parties d'un document XSL-FO .....	62
<u>Créer un document XSL-FO</u> .....	63
Créer et enrichir les blocs de contenu .....	64

Ajouter des images .....	65
Définir un modèle de page .....	66
<u>Créer un en-tête de modèle de page .....</u>	<u>67</u>
Utiliser XSLT pour créer un document XSL-FO ...	68
Insérer des sauts de page .....	69
Produire une sortie en colonnes.....	70
Ajouter un nouveau modèle de page .....	71

## PARTIE 3 - LES DTD

<b><u>CHAPITRE 6. Création d'une DTD .....</u></b>	<b><u>75</u></b>
Utiliser les DTD .....	76
Définir un élément qui contient du texte .....	77
Définir un élément vide .....	78
<u>Définir un élément contenant un fils .....</u>	<u>79</u>
Définir un élément qui contient plusieurs fils ...	80
Définir plusieurs occurrences.....	81
Définir des choix .....	82
<u>Définir un élément pouvant contenir n'importe quoi .....</u>	<u>83</u>
Attributs .....	84
Définir des attributs .....	85
<u>Définir des valeurs par défaut .....</u>	<u>86</u>
<u>Définir des attributs avec des choix .....</u>	<u>87</u>
<u>Définir des attributs ayant des valeurs uniques ..</u>	<u>88</u>
<u>Utiliser les attributs ayant des valeurs uniques ..</u>	<u>89</u>
<u>Limiter les attributs à des noms XML valides...</u>	<u>90</u>
<b><u>CHAPITRE 7. Entités et notations dans les DTD .....</u></b>	<b><u>91</u></b>
<u>Créer une entité générale .....</u>	<u>92</u>
<u>Utiliser les entités générales.....</u>	<u>93</u>
<u>Créer une entité générale externe.....</u>	<u>94</u>
<u>Utiliser les entités générales externes .....</u>	<u>95</u>
<u>Créer des entités pour du contenu non analysé ..</u>	<u>96</u>
<u>Intégrer un contenu non analysé .....</u>	<u>97</u>
Créer et utiliser des entités paramètres .....	98
<u>Créer une entité paramètre externe .....</u>	<u>99</u>

<b>CHAPITRE 8. Validation et utilisation des DTD .....</b>	<b>101</b>
Créer une DTD externe .....	102
Déclarer une DTD externe .....	103
Déclarer et créer une DTD interne.....	104
Valider des documents XML par rapport à une DTD .....	105
Nommer une DTD externe publique .....	106
Déclarer une DTD publique externe .....	107
Avantages et inconvénients des DTD .....	108

## PARTIE 4 - XML SCHEMA

<b>CHAPITRE 9. Introduction à XML Schema ..</b>	<b>111</b>
Utiliser XML Schema .....	112
Créer un XML Schema simple .....	114
Associer un XML Schema à un document XML...115	115
Annoter les schémas .....	116

<b>CHAPITRE 10. Définition des types simples ..</b>	<b>117</b>
Définir un élément de type simple .....	118
Utiliser les types temporels .....	120
Utiliser les types numériques .....	122
Prédéfinir le contenu d'un élément .....	123
Créer des types simples personnalisés .....	124
Créer des types personnalisés nommés .....	125
Indiquer un intervalle de valeurs admises .....	126
Indiquer un ensemble de valeurs admises .....	128
Limiter la longueur d'un élément .....	129
Préciser un motif pour un élément .....	130
Limiter le nombre de chiffres d'un nombre .....	132
Créer un type liste .....	133
Créer un type union .....	134

<b>CHAPITRE 11. Définition des types composés ..</b>	<b>135</b>
Introduction aux types composés.....	136
Créer des types composés anonymes .....	138
Créer des types composés nommés .....	139

Définir des types composés contenant des éléments fils .....	140
Exiger que les éléments fils apparaissent en séquence .....	141
Autoriser un ordre quelconque des éléments fils .....	142
Créer un ensemble de choix.....	143
Définir des éléments ne contenant que du texte..	144
Définir des éléments vides.....	145
Définir des éléments avec un contenu mixte .....	146
Dériver des types composés à partir de types composés existants .....	147
Référencer des éléments définis globalement ...	148
Contrôler le nombre d'occurrences .....	149
Définir des groupes modèles nommés .....	150
Référencer un groupe modèle nommé.....	151
Définir des attributs .....	152
Exiger un attribut .....	153
Prédéfinir le contenu d'un attribut .....	154
Définir des groupes d'attributs.....	155
Référencer un groupe d'attributs .....	156
Définitions locales et globales .....	157

## PARTIE 5 - ESPACES DE NOMS

<b>CHAPITRE 12. Espaces de noms XML.....</b>	161
Concevoir un nom d'espace de noms .....	162
Déclarer un espace de noms par défaut .....	163
Déclarer un préfixe de nom d'espace de noms ..	164
Étiqueter des éléments avec un préfixe d'espace de noms.....	165
Influence des espaces de noms sur les attributs ..	166
<b>CHAPITRE 13. Utilisation des espaces de noms XML .....</b>	167
Remplir un espace de noms XML .....	168
XML Schema, documents XML et espaces de noms .....	169
Référencer les composants d'un XML	

Schema situés dans un espace de noms .....	170
Espaces de noms et validation XML .....	171
Ajouter tous les éléments définis localement....	172
Ajouter des éléments locaux particuliers .....	173
XML Schema réparti dans plusieurs fichiers ...	174
XML Schema avec plusieurs espaces de noms ..	175
Le schéma des schémas comme espace de noms par défaut .....	176
Espaces de noms et DTD .....	177
XSLT et espaces de noms .....	178

## PARTIE 6 - RECOMMANDATIONS RÉCENTES DU W3C

<b>CHAPITRE 14. XSLT 2.0 .....</b>	181
Compléter XSLT .....	182
Créer une feuille de style simplifiée .....	183
Produire des documents XHTML .....	184
Produire plusieurs documents de sortie .....	185
Création de fonctions .....	186
Appeler des fonctions définies par l'utilisateur ..	187
Grouper les résultats en fonction de valeurs communes.....	188
Valider le résultat de XSLT .....	189

<b>CHAPITRE 15. XPATH 2.0 .....</b>	191
XPath 1.0 et XPath 2.0 .....	192
Moyenne des valeurs d'une séquence .....	194
Trouver les valeurs minimale et maximale .....	195
Formatage des chaînes de caractères .....	196
Tester des conditions .....	197
Quantifier une condition .....	198
Supprimer les éléments dupliqués .....	199
Parcourir des séquences .....	200
Utiliser la date et l'heure courantes .....	201
Placer des commentaires .....	202
Traiter un texte source non XML .....	203

<b>CHAPITRE 16. XQUERY 1.0 .....</b>	205
XQuery 1.0 vs. XSLT 2.0 .....	206
Composer un document XQuery .....	207
Identifier un document source XML .....	208
Utiliser des expressions XPath.....	209
Utiliser des expressions FLWOR.....	210
Tests et expressions conditionnelles .....	212
Joindre deux sources de données apparentées ..	213
Créer et appeler des fonctions utilisateur .....	214
XQuery et les bases de données .....	215

## PARTIE 7 - XML EN PRATIQUE

<b>CHAPITRE 17. Ajax, RSS, SOAP, etc. .....</b>	219
Introduction à Ajax .....	220
Exemples Ajax.....	222
Introduction à RSS.....	224
Schéma de RSS .....	225
Étendre RSS .....	226
SOAP et services web .....	228
Schéma d'un message SOAP .....	229
WSDL .....	230
Introduction à KML.....	232
Fichier KML simple .....	233
ODF et OOXML .....	234
eBooks, ePub, etc. ....	236
Outils pour XML en pratique .....	238

## ANNEXES

<b>ANNEXE A. Outils XML .....</b>	243
Éditeurs XML .....	244
Autres éditeurs XML .....	246
Outils et ressources XML .....	247
<b>ANNEXE B. Encodage des caractères et entités .....</b>	249
Préciser l'encodage des caractères .....	250
Utiliser des références numériques de caractères (NCR) .....	251
Utiliser des références d'entités .....	252
Caractères Unicode .....	253
<b>ANNEXE C. Lexique anglais/français .....</b>	255
<b>ANNEXE D. Lexique français/anglais .....</b>	257
<b>ANNEXE E. Mots-clés .....</b>	259
Éléments de syntaxe XML.....	259
Éléments de syntaxe XSD .....	260
Éléments de syntaxe XSL .....	261
Éléments de syntaxe XPath .....	261
<b>Index.....</b>	263

*image  
not  
available*

*image  
not  
available*

```
x m 1  
<?xml version="1.0"?>  
<anciennes_merveilles>  
  <merveille>  
    <nom langue="Français">Le colosse de  
      Rhodes</nom>  
    <nom langue="Grec">Κολοσσός της Ρόδου</nom>  
    <lieu>Rhodes en Grèce</lieu>  
    <hauteur unité="mètres">33</hauteur>  
    <image_principale fichier="colosse.jpg"  
      l="528" h="349"/>  
    <source idsection="101" idjournal="21"/>  
  </merveille>  
  ...  
</anciennes_merveilles>
```

**Figure i.2** Bien que XML ne semble pas bien différent de HTML, vous remarquerez que les balises ne sont pas les mêmes que celles de HTML et qu'elles décrivent le contenu qu'elles délimitent. La syntaxe d'un document XML est également plus stricte que celle d'une page HTML – nous présenterons ces règles au Chapitre 1.

## Puissance de XML

Pourquoi utiliser XML ? Qu'apporte-t-il aux techniques et aux langages existants ? En fait, XML a été spécifiquement conçu pour le stockage et le transport des données. Il ressemble beaucoup à HTML puisqu'il utilise également des balises, des attributs et des valeurs (voir Figure i.2) mais, au contraire de ce dernier, il ne sert pas à afficher des informations.

Une autre raison d'utiliser XML est qu'il est très facile de l'étendre et de l'adapter à ses besoins. Grâce à lui, on peut concevoir son propre langage à balises, qui servira ensuite à stocker nos informations – ce langage personnalisé contiendra les balises qui décrivent les données qu'elles délimitent et ces balises pourront ensuite être réutilisées dans d'autres applications XML, réduites ou augmentées en fonction des besoins.

XML permet également d'échanger des données entre des systèmes et des organisations car un document XML n'est, finalement, qu'un simple fichier texte. Il est bien structuré, simple à comprendre, facile à traiter et à manipuler : c'est un document "lisible", comme vous avez pu le constater avec les exemples des Figures i.1 et i.2.

Enfin, XML est une spécification non propriétaire et tout le monde peut donc l'utiliser librement. Il a été créé par le W3C ([www.w3.org/](http://www.w3.org/)), un consortium international consacré au développement des normes et des spécifications liées au Web. Cette ouverture permet à n'importe quelle organisation, quelle que soit sa taille, d'utiliser XML pour le partage des informations. En outre, une communauté internationale a permis de créer de nouvelles applications reposant sur XML, permettant ainsi de franchir les barrières commerciales érigées par les standards propriétaires et les règles gouvernementales.

*image  
not  
available*

# 1

## Créer des documents XML

La spécification XML définit comment écrire un document au format XML. XML n'est pas un langage en lui-même mais, en revanche, un document XML est écrit dans un langage à balises spécifique respectant la spécification XML. Il peut, par exemple, exister des langages spécifiques permettant de décrire des données généalogiques, chimiques ou commerciales, grâce auxquels vous pouvez créer vos propres documents XML.

Chaque langage à balises spécifique créé à partir de la spécification XML doit respecter la grammaire sous-jacente de XML ; c'est donc par là que nous commencerons ce livre. Dans ce chapitre, vous apprendrez les règles d'écriture des documents XML, quel que soit le langage à balises que vous utiliserez ensuite.

Officiellement, les langages à balises spécifiques créés avec XML sont appelés *applications XML*. En d'autres termes, ces langages – XSLT, RSS, SOAP, etc. – sont des applications de XML. Cependant, pour moi, une application est plutôt un programme logiciel complet – comme Photoshop – et je trouve donc ce terme si peu précis que j'éviterai de l'utiliser.

### Outils de création des documents XML

Comme pour HTML, vous pouvez utiliser n'importe quel éditeur ou traitement de texte pour écrire du XML. Il existe également de nombreux éditeurs spécialisés, dotés de fonctionnalités spécifiques, comme la validation automatique en cours de frappe (voir Annexe A).

Je supposerai désormais que vous savez créer de nouveaux documents, en ouvrir d'anciens et les sauvegarder. Assurez-vous simplement que tous vos documents XML sont sauvegardés avec l'extension `.xml`.

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

```
x m l  
<?xml version="1.0"?>
```

**Figure 1.10** La déclaration XML étant une instruction de traitement, pas un élément, elle n'a pas de balise fermante.

## Commencer un document XML

En principe, un document XML doit commencer par une déclaration indiquant la version de XML utilisée. Cette ligne s'appelle la *déclaration XML*.

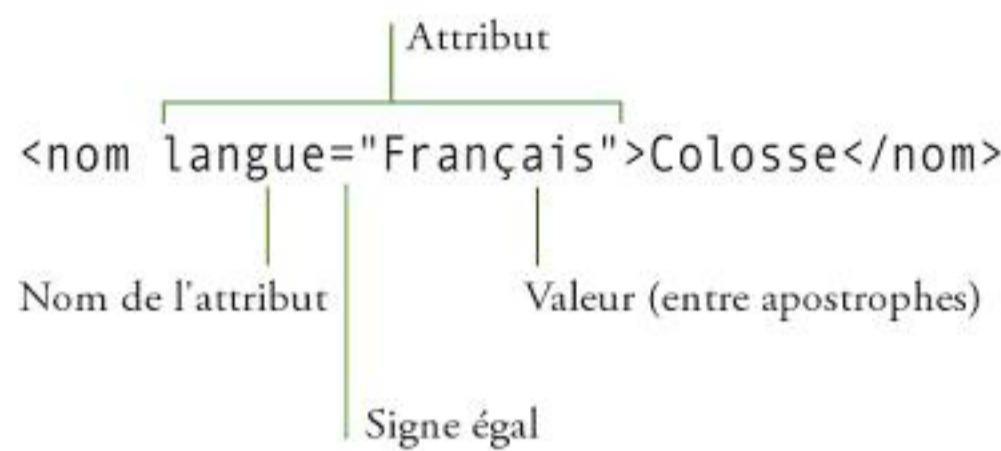
### CONSEILS

- *Le W3C a publié en 2006 une recommandation pour XML version 1.1 mais elle apporte peu de choses par rapport à la version 1.0 et elle n'est quasiment pas supportée par les outils existants.*
- *Assurez-vous de placer le numéro de version entre apostrophes doubles ou simples (peu importe, du moment qu'elles sont appariées).*
- *Les balises commençant par <? et se terminant par ?> sont appelées instructions de traitement. Outre la déclaration de la version de XML, ces instructions permettent également de préciser la feuille de style à utiliser, par exemple. Les feuilles de style seront étudiées dans la Partie 2 de ce livre.*
- *Cette instruction de traitement XML peut également préciser l'encodage des caractères du document (UTF-8, ISO-8859-1, etc.). Les encodages des caractères sont présentés à l'Annexe B.*

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*



**Figure 1.16** La valeur d'un attribut doit être entourée d'apostrophes simples ou doubles.

```
x m 1
<?xml version="1.0"?>
<anciennes_merveilles>
  <merveille>
    <nom langue="Français">Colosse de Rhodes</nom>
    <nom langue="Grec">Κολοσσός της Ρόδου</nom>
    <lieu>Rhodes en Grèce</lieu>
    <hauteur unité="mètres">32</hauteur>
  </merveille>
</anciennes_merveilles>
```

**Figure 1.17** Attributs permettant d'ajouter des informations sur le contenu d'un élément.

## Attributs

Un attribut permet de stocker des informations supplémentaires sur un élément sans ajouter de texte au contenu de l'élément lui-même. Il est formé d'une paire nom/valeur et est ajouté à la balise ouvrante d'un élément, comme le montre la Figure 1.16.

### CONSEILS

- *Les noms des attributs suivent les mêmes règles que celles des noms d'éléments.*
- *Pour un même élément, tous les attributs doivent porter un nom différent.*
- *À la différence de HTML, les valeurs des attributs doivent impérativement être placées entre apostrophes simples ou doubles.*
- *Si la valeur d'un attribut contient des apostrophes doubles, il faut entourer cette valeur d'apostrophes simples (et vice versa).*  
*Par exemple : commentaire= 'Elle a dit "Le Colosse est tombé !"'.*
- *Comme en programmation, une bonne pratique consiste à considérer les attributs comme des métadonnées, c'est-à-dire des données sur les données. En d'autres termes, les attributs devraient servir à stocker des informations sur le contenu de l'élément, pas le contenu lui-même. C'est ce que l'on fait à la Figure 1.17.*
- *Un autre moyen de marquer et d'identifier des informations distinctes consiste à utiliser des éléments imbriqués, comme nous l'avons vu précédemment.*

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

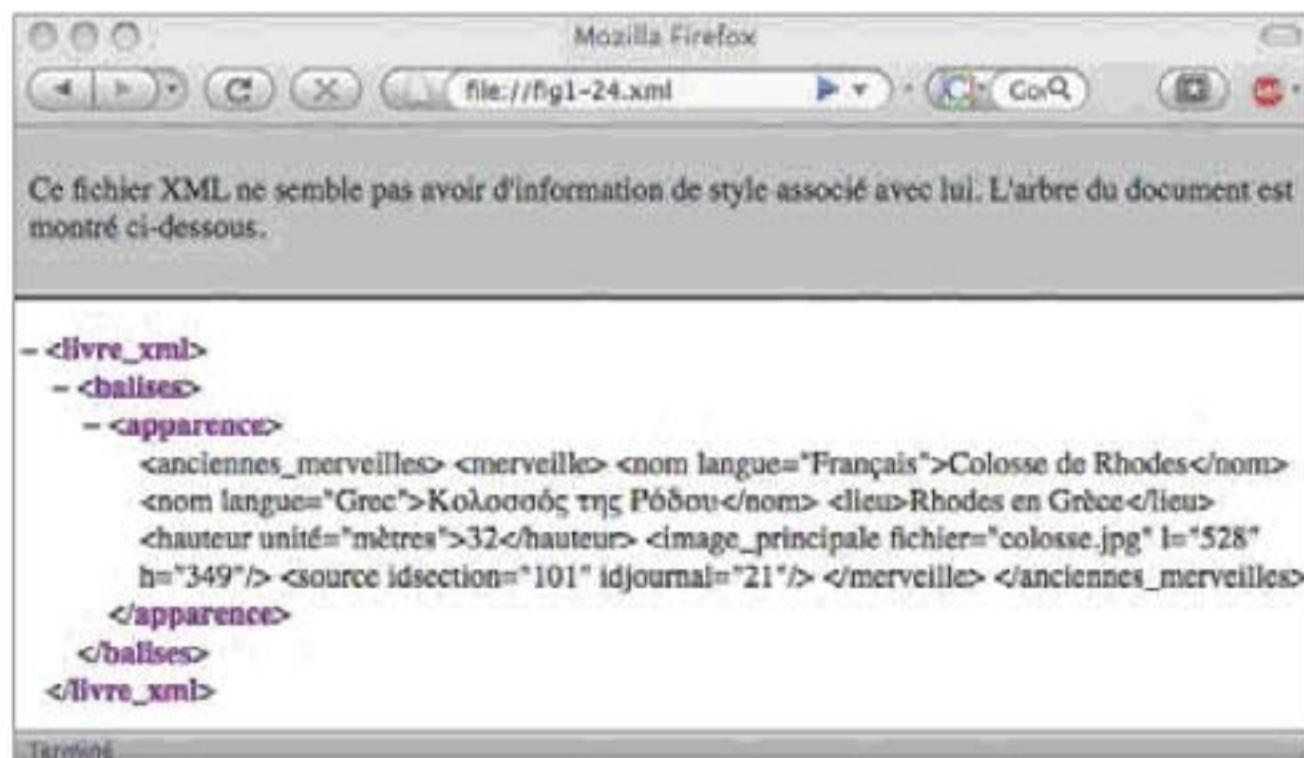
```

x m l

<?xml version="1.0"?>
<livre_xml>
  <balises>
    <apparence>
      <![CDATA[
        <anciennes_merveilles>
          <merveille>
            <nom langue="Français">Colosse de Rhodes</nom>
            <nom langue="Grec">Κολοσσός της Ρόδου</nom>
            <lieu>Rhodes en Grèce</lieu>
            <hauteur unité="mètres">32</hauteur>
            <image_principale fichier="colosse.jpg" l="528" h="349"/>
            <source idsection="101" idjournal="21"/>
          </merveille>
        </anciennes_merveilles>
      ]]>
    </apparence>
  </balises>
</livre_xml>

```

**Figure 1.23** La section CDATA sera simplement affichée, sans que le processeur XML ne la traite d'abord.



**Figure 1.24** L'affichage de ce document avec Firefox montre que les éléments contenus dans la section CDATA sont traités comme du texte alors que les balises livre\_xml, balises et apparence sont traitées par le processeur XML.

## Affichage des éléments sous forme de texte

Si vous voulez placer des éléments et des attributs XML et faire en sorte que le processeur les affiche simplement au lieu de les interpréter, vous devez les placer dans une section CDATA, comme à la Figure 1.23.

### CONSEILS

- Les sections CDATA servent également souvent à insérer du code HTML ou JavaScript tout en évitant que le processeur XML ne les traite.
- CDATA signifie (unparsed) Character Data (données textuelles non traitées). C'est donc l'opposé de PCDATA, qui signifie Parsed Character Data (données textuelles traitées) et sera présenté au Chapitre 6.
- Dans une section CDATA, la signification spéciale des symboles est ignorée. Pour afficher le signe inférieur et l'esperluette, il faut donc écrire < et & : si vous écriviez &lt; et &amp;, ces entités ne seraient pas remplacées par les symboles qu'elles représentent.
- Les sections CDATA ne peuvent pas être imbriquées.
- Une section CDATA peut apparaître n'importe où dans l'élément racine d'un document XML.
- Si, pour une raison ou pour une autre, vous devez écrire ]]> sans pour autant fermer une section CDATA, représentez le signe > par son entité, &gt;. Reportez-vous à l'Annexe B pour plus de détails sur l'écriture des caractères spéciaux.

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

# 2

## XSLT

Maintenant que vous connaissez le langage XML et que vous savez créer et lire des documents XML, l'étape suivante consiste à les formater. Les détails de mise en forme des documents XML se trouvaient initialement dans une spécification nommée XSL (*eXtensible Style Language*). Celle-ci prenant trop de temps à être mise en place, le W3C l'a divisée en deux parties : XSLT (*Transformations*) et XSL-FO (*Formatting Objects*).

Ce chapitre et les deux suivants expliquent comment utiliser XSLT pour transformer des documents XML afin de produire un autre document XML ou une page HTML mais, en réalité, vous pouvez obtenir pratiquement n'importe quel type de document avec une transformation XSL.

La transformation d'un fichier XML implique d'utiliser XSLT pour analyser son contenu, puis d'effectuer certaines opérations en fonction des éléments qu'il contient. Vous pouvez ainsi organiser le résultat obtenu selon certains critères, n'en afficher que certaines parties et bien plus encore.

On utilise généralement XSL-FO pour produire un résultat imprimable, comme un document PDF. Comme elle n'est pas reconnue par les navigateurs, cette transformation nécessite un logiciel particulier, comme nous le verrons au Chapitre 5.

La plupart des exemples de cette partie du livre utilisent un unique fichier XML et plusieurs fichiers XSLT, chacun étant généralement construit à partir du précédent. Je vous conseille fortement de télécharger les exemples à partir de la page web consacrée à cet ouvrage, sur le site [www.pearson.fr](http://www.pearson.fr).

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

x s l t
<?xml version="1.0"?> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/ Transform" version="1.0"> <xsl:template match="/">  </xsl:template> </xsl:stylesheet>

**Figure 2.6** La règle racine (`match=" / "`) est le point de départ de tout traitement XSLT.

## Création de la règle racine

La règle racine est la première chose que recherche le processeur XSLT dans une feuille de style. Cette règle définit l'ensemble des règles qui s'appliquent au nœud racine du document XML. Plus précisément, elle décrit comment traiter ou transformer le contenu du nœud racine pour produire un nouveau résultat.

Comme le montre la Figure 2.6, la règle racine est comprise entre `<xsl:template match=" / ">` et `</xsl:template>`.

### CONSEILS

- *Bien que le processeur XSLT ne tienne pas compte de l'endroit où apparaît la règle racine dans la feuille de style XSLT, il est plus lisible (pour vous et ceux qui seront amenés à relire cette feuille) de la placer au tout début.*
- *Toutes les transformations XSLT doivent commencer par la règle racine. Si une feuille de style n'en contient pas, le processeur XSLT utilisera automatiquement une règle racine par défaut. Celle-ci se contente d'énumérer en texte simple toutes les données du document XML, ce qui n'est probablement pas ce que vous souhaitez obtenir.*

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

```

h t m l

<html>
  <head>
    <meta http-equiv="Content-Type"
          content="text/html;
          charset=UTF-8">
    <title>Merveilles du monde</title>
  </head>
  <body>
    <h1>Les sept merveilles du monde antique
    </h1>
    <p>Le colosse de Rhodes est l'une de ces
       merveilles.</p>
  </body>
</html>

```

**Figure 2.12** Lorsque le processeur XSLT applique la règle racine de la Figure 2.11, il produit d'abord les en-têtes HTML. Puis, lorsqu'il arrive à l'élément `xsl:value-of`, il ne produit que la valeur du premier nœud qu'il trouve, soit Colosse de Rhodes. Nous verrons plus loin comment renvoyer plusieurs valeurs.



**Figure 2.13** Le processeur XSLT utilise maintenant les informations qui se trouvent dans le document source XML.

## CONSEILS

- Comme nous le verrons dans la section "Déterminer le nœud courant", page 40 vous pouvez utiliser `select=". "` pour produire le contenu du nœud courant.
- Si l'expression `select` capture plusieurs nœuds du document XML, seul le premier sera pris en compte. Dans l'exemple de la Figure 2.12, plusieurs nœuds nom correspondaient mais la transformation n'a produit que la valeur du premier (Colosse de Rhodes).
- Si l'on veut agir sur plusieurs nœuds, il faut utiliser un autre élément XSLT, qui sera décrit à la section "Boucles sur les nœuds", page 28.
- À la Figure 2.11, si l'on avait voulu produire les nœuds nom dont l'attribut `langue` vaut "Grec", il aurait fallu écrire `xsl:value-of select="nom[@langue='Grec']"`. Pour plus de détails, voir la section "Traitement conditionnel des nœuds", page 30.
- Si l'expression `select` capture un nœud, le résultat est la valeur textuelle de ce nœud (c'est-à-dire le texte qu'il contient). Si ce nœud a des éléments fils, le résultat contiendra également le texte de ces éléments.
- Si l'expression `select` capture un nœud vide, rien n'est produit.
- Si l'expression `select` produit un nombre, celui-ci est converti en chaîne avant d'être ajouté au résultat.
- Si l'expression `select` produit une expression booléenne – logique – (valant soit `true` soit `false`), le résultat produit sera soit le texte "`true`", soit le texte "`false`".

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

```
x s l t

...
<xsl:choose>
  <xsl:when test="hauteur != 0">
    <xsl:value-of select="hauteur" />
  </xsl:when>
  <xsl:otherwise>
    inconnue
  </xsl:otherwise>
</xsl:choose>
...

```

**Figure 2.20** Dans le document XML, la hauteur des jardins suspendus de Babylone est arbitrairement fixée à zéro, uniquement pour avoir une valeur numérique pour ce nœud car, en réalité, on ne sait même pas s'ils ont existé.

```
h t m l

...
<tr>
  <td>
    <strong>Grande pyramide de Kheops</strong>
  </td>
  <td>Gizeh en Égypte</td>
  <td>138</td>
</tr>
<tr>
  <td>
    <strong>Jardins suspendus de Babylone
    </strong>
  </td>
  <td>Al Hillah en Irak</td>
  <td>
    inconnue
  </td>
</tr>
...

```

**Figure 2.21** Grâce à l'élément `xsl:choose` de la Figure 2.20, on peut afficher le mot `inconnue` au lieu d'une valeur nulle pour la hauteur des jardins suspendus de Babylone (comme à la Figure 2.19).

## Utilisation de plusieurs choix

L'élément `xsl:if` décrit dans la section précédente ne permet de tester qu'une seule condition et n'autorise qu'une seule action. Si vous voulez tester plusieurs conditions différentes, utilisez `xsl:choose` afin de réagir à chacune d'elles. Son utilisation la plus simple consiste à effectuer une certaine opération lorsque la condition est vraie et une autre lorsqu'elle est fausse.

L'élément `xsl:choose` contient plusieurs éléments `xsl:when` et, éventuellement, un élément `xsl:otherwise`. Comme pour `xsl:if`, l'attribut `test` de chaque élément `xsl:when` contient une expression désignant un ensemble de nœuds, une chaîne ou un nombre : l'élément n'effectuera son traitement que si cette expression ne renvoie pas un résultat vide ou nul. S'il est présent, l'élément `xsl:otherwise` n'effectuera son traitement que si aucune des expressions des éléments `xsl:when` n'est vérifiée. Consultez le Chapitre 3 pour plus de détails sur l'écriture de ces expressions.

### CONSEIL

- *Lorsqu'il y a plusieurs conditions, dès qu'une condition est considérée comme vraie, toutes les suivantes sont ignorées (même si elles sont également vraies). L'action contenue dans la première condition vraie est donc la seule qui sera réalisée.*

Nom de la merveille	Lieu	Hauteur
Colosse de Rhodes (Κολοσσός της Ρόδου)	Rhodes en Grèce	32
Grande pyramide de Khéops	Gizeh en Égypte	138
Jardins suspendus de Babylone	Al Hillah en Irak	inconnue
Statue de Zeus à Olympie (Δίας ψυθολογία)	Olympie en Grèce	12
Temple d'Artémis à Éphèse (Ἄρτεμιν)	Éphèse en Turquie	18
Mausolée d'Halicarnasse (Μαυσωλεὺς Αἱλικαρνασσού)	Bodrum en Turquie	41
Phare d'Alexandrie (ὁ Φάρας τῆς Ἀλεξανδρείας)	Alexandrie en Égypte	117

**Figure 2.22** L'information est maintenant plus judicieuse.

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

```

x s l t

...
<h2>Historique</h2>
<xsl:for-each select="anciennes_merveilles/
merveille">
  <xsl:sort select="hauteur"
             order="descending"
             data-type="number" />
  <a><xsl:attribute name="nom">
    <xsl:value-of select="nom[@langue=
      'Français']"/>
  </xsl:attribute>
  </a>
  <xsl:value-of select="nom[@langue=
    'Français']"/>
  <xsl:apply-templates
    select="nom[@langue!='Français']"/>
  <br /><br />
</xsl:for-each>

```

**Figure 2.31** Dans cet extrait de la même feuille de style, on a ajouté une nouvelle section intitulée Historique pour afficher l'histoire des anciennes merveilles. On réutilise la règle créée à la Figure 2.29 en l'appelant avec l'élément `xsl:apply-templates` pour afficher le nom non français.



**Figure 2.32** En déplaçant le traitement des langues non françaises dans une règle à part, on peut réutiliser le même code et produire deux fois les mêmes informations sans devoir réécrire ces instructions.

Il faut maintenant utiliser les règles que l'on a créées, c'est-à-dire savoir comment appliquer une règle à un nœud particulier du document XML. Pour ce faire, on utilise `xsl:apply-templates` afin de contrôler où et quand la transformation décrite par la règle sera utilisée dans le document final.

Cette instruction comprend un attribut `select` contenant une expression permettant d'identifier le(s) nœud(s) du document XML dont il faut appliquer les règles. Reportez-vous au Chapitre 3 pour plus de détails sur la syntaxe de ces expressions.

## CONSEILS

- Si votre feuille de style contient plusieurs règles, l'ordre des éléments `xsl:apply-templates` détermine l'ordre de traitement des règles.
- Si vous ne précisez pas l'attribut `select` de `xsl:apply-templates`, le processeur recherche et applique une règle à chaque fils du nœud courant.
- Si vous utilisez `xsl:apply-templates` et qu'il n'existe aucune règle correspondant au nœud courant, le processeur XSLT utilise automatiquement une règle prédéfinie. Pour le nœud racine ou un nœud élément, il recherche une règle correspondant à chaque nœud fils. Pour un nœud texte ou attribut, il produit la valeur du nœud sous forme textuelle.
- Un élément `xsl:apply-templates` peut contenir un élément `xsl:sort`.
- Si vous ne l'avez pas encore fait, je vous conseille de télécharger et de relire les fichiers des exemples.

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

x m l
<anciennes_merveilles> <merveille> <nom langue="Français">Le colosse de Rhodes</nom> <nom langue="Grec">Κολοσσός της Ρόδου</nom> <lieu>Rhodes en Grèce</lieu> <hauteur unité="mètres">32</hauteur> <historique>...</historique> <image_principale ... /> <source ... /> </merveille> </anciennes_merveilles>

**Figure 3.2** Document XML représenté par l’arborescence de la Figure 3.1.

Un *chemin d'accès absolu* commence par / (barre de fraction), éventuellement suivi d'un chemin relatif. Un / seul permet de sélectionner le nœud racine du document ; s'il est suivi d'un chemin d'accès relatif, le chemin est un chemin relatif qui part du nœud racine.

Le choix entre un chemin relatif ou absolu dépend des situations. On utilise le plus souvent les premiers car ils produisent un ensemble de nœuds par rapport au nœud courant, ce qui est généralement ce dont on a besoin.

## Utilisation des nœuds trouvés

Lorsque l'on utilise des chemins d'accès, on emploie souvent le nœud ou l'ensemble des nœuds trouvés comme un conteneur des autres éléments à traiter.

Parfois, l'on a besoin de connaître la valeur du nœud. XPath distingue sept types de nœuds et permet de récupérer la valeur de chacun d'eux : les nœuds racines (il n'y en a qu'un et un seul), les nœuds éléments, les nœuds textuels, les nœuds attributs, les noeuds commentaires, les nœuds d'instructions de traitement et les nœuds d'espaces de noms. Pour certains nœuds, cette valeur fait partie du nœud lui-même ; pour d'autres, elle dépend de celle de ses nœuds descendants.

## CONSEILS

- *La syntaxe du langage XPath s'inspire de celles du système de fichiers Unix "chemin/fichier".*
- *Le nœud courant est l'élément, ou nœud, en cours de traitement, tandis que le nœud contextuel est celui d'où part le chemin d'accès. Dans la plupart des cas, ces deux termes sont interchangeables et c'est la raison pour laquelle j'utiliserai le terme nœud courant dans ce livre.*

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

```
x m 1

...
<merveille>
  <nom langue="Français">Le colosse de Rhodes
  </nom>
  <nom langue="Grec">Κολοσσός της Ρόδου</nom>
...
<historique>
  <année_construction ère="BC">282</année_
  construction>
  <année_destruction ère="BC">226</année_
  destruction>
  <motif_destruction>tremblement de terre
  </motif_destruction>
  <histoire>En 294 avant J. -C., ...
  </histoire>
</historique>
```

**Figure 3.8** Les éléments *historique* et *nom* sont tous les deux des fils de l'élément *merveille* et sont donc frères.

```
x s l t

...
<xsl:template match="historique">
...
<xsl:value-of select="..//nom[@langue=
'Français']"/>
<xsl:apply-templates select="..
  nom[@langue!='Français']"/>
a été construit(e) en <xsl:value-of
select="année_construction"/>
...
```

**Figure 3.9** Lorsque le processeur applique cette règle *historique*, *historique* devient le nœud courant. Si l'on veut alors désigner l'élément *nom* (qui est son frère), il faut remonter d'un niveau avec .. (vers l'élément père, *merveille*), puis utiliser / pour passer vers un autre fils et, enfin, nom pour préciser l'élément fils.

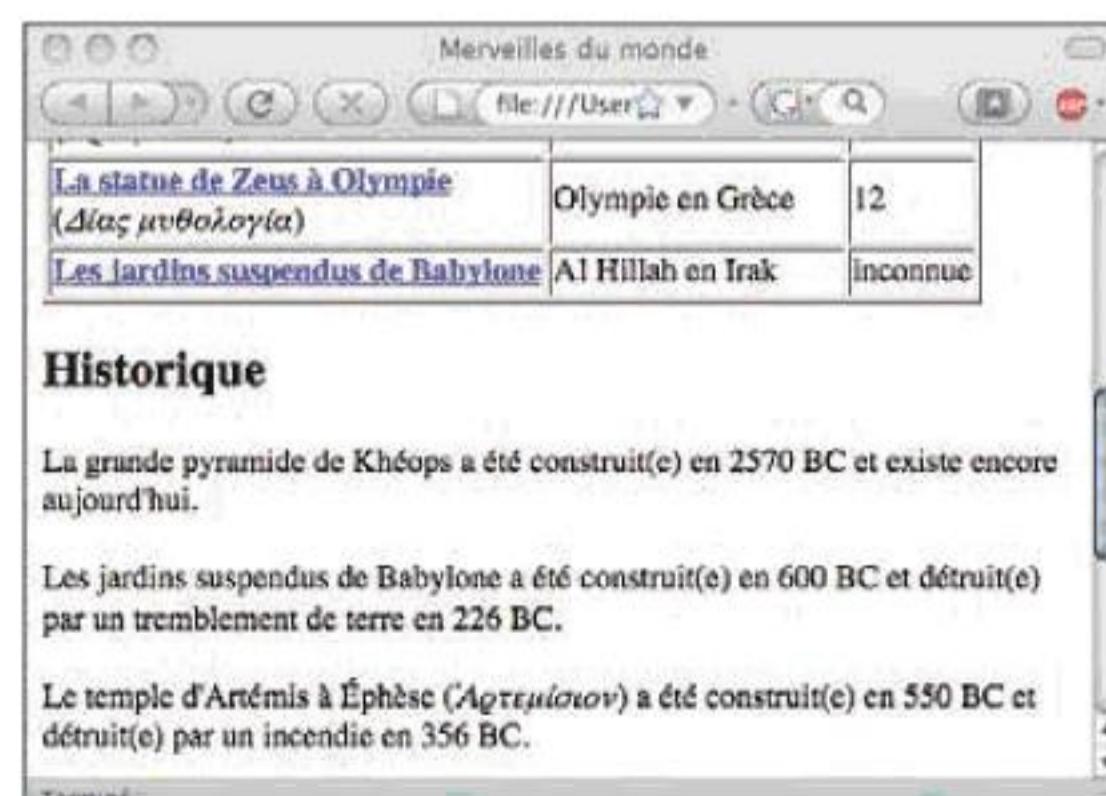
## Sélectionner le père ou le frère d'un nœud

Si la relation entre le nœud courant et le nœud cible est assez claire, il est bien plus simple d'utiliser un raccourci plutôt qu'un chemin absolu partant du nœud racine.

1. Pour sélectionner le nœud père du nœud courant, tapez .. (deux points).
2. Pour sélectionner le frère d'un nœud, revenez au nœud père en utilisant .., puis ajoutez /frère, où *frère* désigne le nœud frère que vous voulez atteindre. Ce frère est donc un fils du père du nœud courant, mais ce n'est pas le nœud courant lui-même.
3. Si nécessaire, vous pouvez ajouter /neveu, où *neveu* désigne un nœud qui est le fils du frère du nœud courant. Vous pouvez répéter cette dernière étape autant de fois que nécessaire pour accéder aux petits-cousins, arrière-petits-cousins et ainsi de suite.

### CONSEILS

- Le symbole .. est souvent combiné avec un attribut afin de trouver l'attribut du nœud père (../@attribut).
- L'astérisque peut servir de caractère joker dans un chemin d'accès. Par exemple, ../\* sélectionne tous les éléments fils du père du nœud courant, y compris le nœud courant lui-même.



**Figure 3.10** Bien que le processeur soit maintenant dans la règle *historique* (et non *merveille*), le résultat est correct tant que l'on précise la nouvelle relation.

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

```
x m l

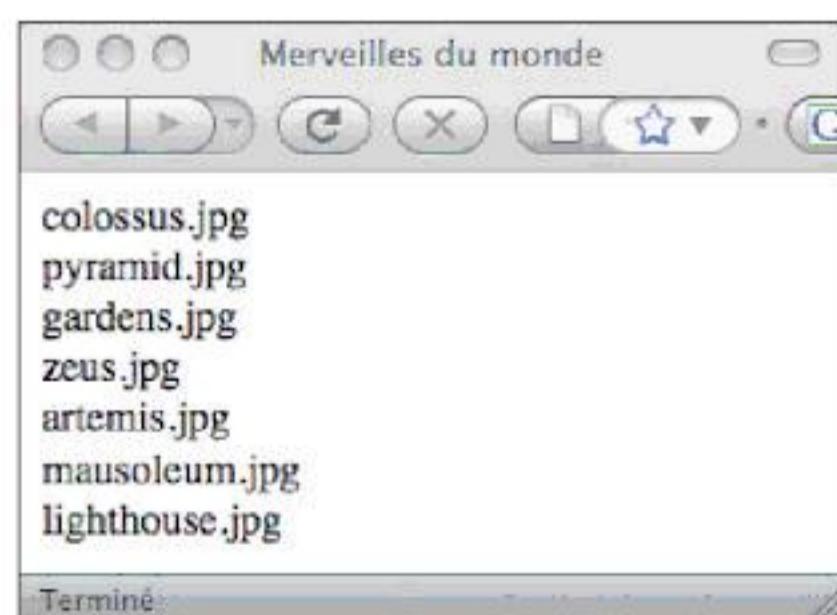
...
<merveille>
  <nom langue="Français">Le phare
  d'Alexandrie</nom>
...
  <image_principale fichier="lighthouse.jpg"
  l="528" h="349"/>
  <source idsection="112" idjournal="53" />
</merveille>
...
```

**Figure 3.20** Certains éléments du document XML contiennent des noms de fichiers d'images.

```
x s l t

...
<xsl:template match="/">
  <html><head><title>Merveilles du monde
  </title></head>
  <body>
    <xsl:apply-templates select="/*/@
      fichier" />
  </body>
</html>
</xsl:template>
...
<xsl:template match="/*/@fichier" >
  <xsl:value-of select=". /><br />
</xsl:template>
```

**Figure 3.21** Les attributs *select* et *match* en gras renvoient tous les nœuds qui ont un attribut *fichier*, où qu'ils se trouvent dans le document source (cette feuille de style est totalement différente des précédentes).



**Figure 3.22** Tous les noms de fichiers apparaissant dans le document source sont affichés, quel que soit leur emplacement dans l'arborescence.

## Sélectionner tous les descendants

Le symbole `//` (double barre de fraction) permet de sélectionner tous les descendants d'un nœud donné (voir Figure 3.21). Comme la plupart des autres raccourcis de ce chapitre, vous pouvez l'utiliser dans un chemin d'accès absolu ou relatif.

- Pour sélectionner tous les descendants du nœud racine, utilisez `//` (deux barres de fraction).
- Pour sélectionner tous les descendants du nœud courant, utilisez `.//` (point et deux barres de fraction).
- Pour sélectionner tous les descendants de n'importe quel nœud, utilisez les techniques des pages précédentes pour accéder au nœud concerné, puis ajoutez `//` à ce chemin.
- Pour sélectionner certains descendants de n'importe quel nœud, faites comme précédemment puis ajoutez le nom des descendants qui vous intéressent.

### CONSEIL

- Pour accéder à un nœud dont vous ne connaissez pas l'emplacement dans le document (ou dont vous ne vous souciez pas), utilisez une expression comme `//nom_élément` pour produire tous les éléments *nom\_élément* du document source, où qu'ils soient.

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

```

x s l t

...
<p>Ces merveilles antiques sont
<xsl:for-each
  select="anciennes_merveilles/merveille/
  nom[@langue='Français']">

  <xsl:value-of select=". " />
  <xsl:choose>

    <xsl:when test="position()=last()">
    </xsl:when>
    <xsl:when test="position()=last()-1" and
    </xsl:when>
    <xsl:otherwise>, </xsl:otherwise>

  </xsl:choose>

  </xsl:for-each>
</p>
...

```

**Figure 4.3** On utilise ici la fonction `position()` pour formater une phrase qui énumère les merveilles. Le nom de la merveille est toujours produit ; s'il s'agit du dernier nom de la liste, on le fait suivre d'un point ; si c'est l'avant-dernier (`position() = last()-1`), on le fait suivre d'une espace, du mot "et" et d'une espace. Sinon on le fait simplement suivre d'une virgule et d'une espace.

## Tester la position

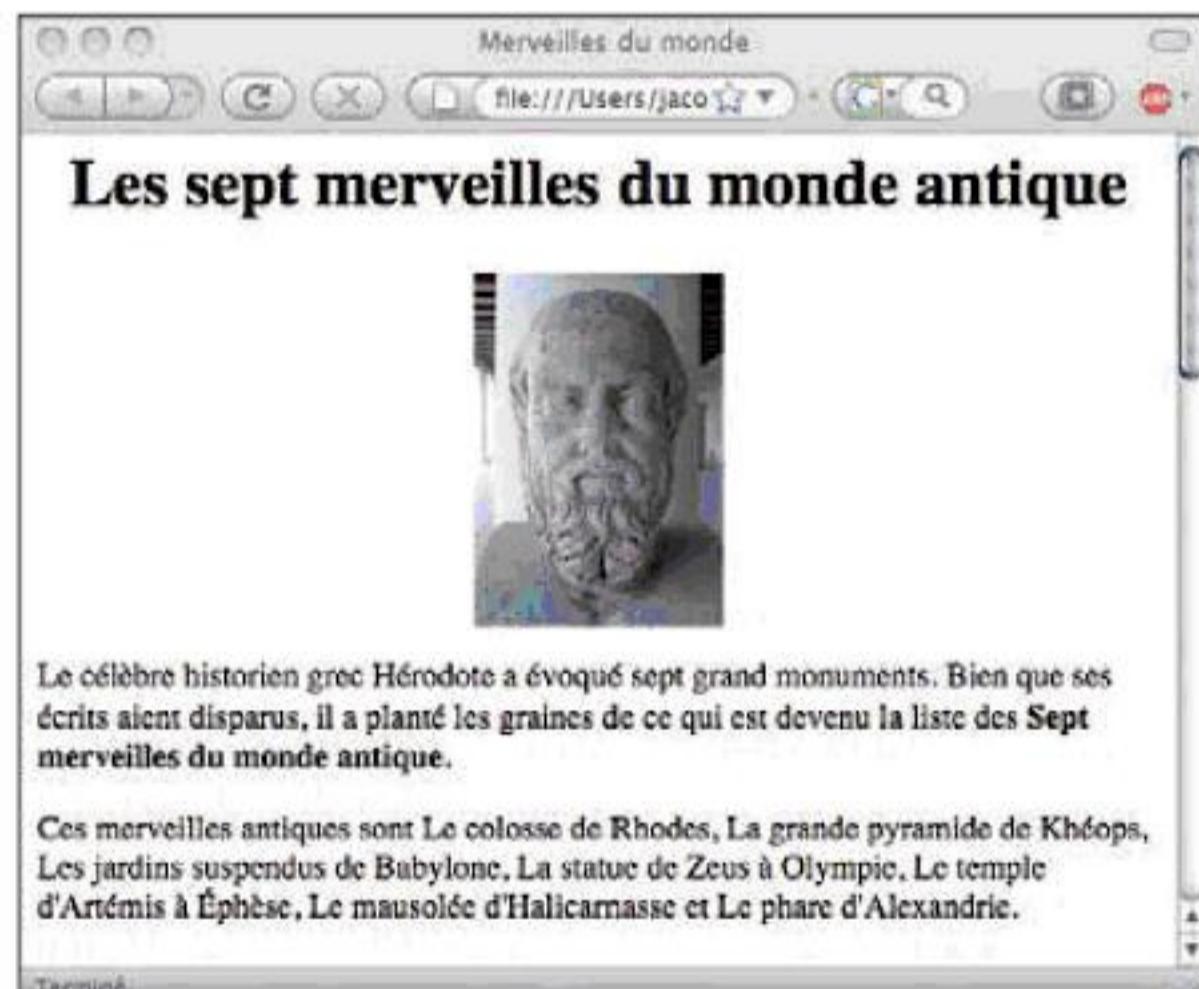
Outre tester les chemins d'accès, vous pouvez choisir un nœud spécifique dans un ensemble de nœuds : le premier, le second... ou même le dernier.

Pour tester la position d'un nœud, utilisez `position() = n`, où `n` est un nombre qui identifie la position du nœud dans l'ensemble de nœuds courant (voir Figure 4.3).

Pour trouver le dernier nœud d'un ensemble, utilisez la fonction `last()`.

### CONSEILS

- Les parenthèses des fonctions `position()` et `last()` doivent être vides.
- Dans un prédictat (une expression booléenne entre crochets permettant de tester une condition), la valeur numérique `n` est un raccourci pour `position()=n`. Ainsi, `merveille[1]` renvoie le premier nœud `merveille`. Vous pouvez utiliser ce raccourci dans le traitement d'une règle, mais pas dans les expressions de test de `xsl:if` ou `xsl:when` ni dans une instruction `xsl:value-of`.



**Figure 4.4** Le résultat de la transformation avec le style de la Figure 4.3 affiche la liste des sept merveilles, correctement formatée.

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

```
x m l
<image_principale fichier="artemis.jpg"
l="528" h="349" />
```

**Figure 4.13** Dans cet extrait, vous pouvez constater que la taille de l'image fait 528 par 349 pixels.

```
x s l t
...
<img>
  <xsl:attribute name="src">
    <xsl:value-of select="./@fichier" />
  </xsl:attribute>
  <xsl:attribute name="width">
    <xsl:value-of select="ceiling(.@l div 2)" />
  </xsl:attribute>
  <xsl:attribute name="height">
    <xsl:value-of select="ceiling(.@h div 2)" />
  </xsl:attribute>
</img>
...
```

**Figure 4.14** Les images seront réduites de moitié dans la page HTML. Comme les attributs width et height n'acceptent que des valeurs entières, on utilise la fonction ceiling() pour arrondir le résultat de la division au plus petit entier supérieur.

```
h t m l
...
<p align="center">
  
</p>
...
```

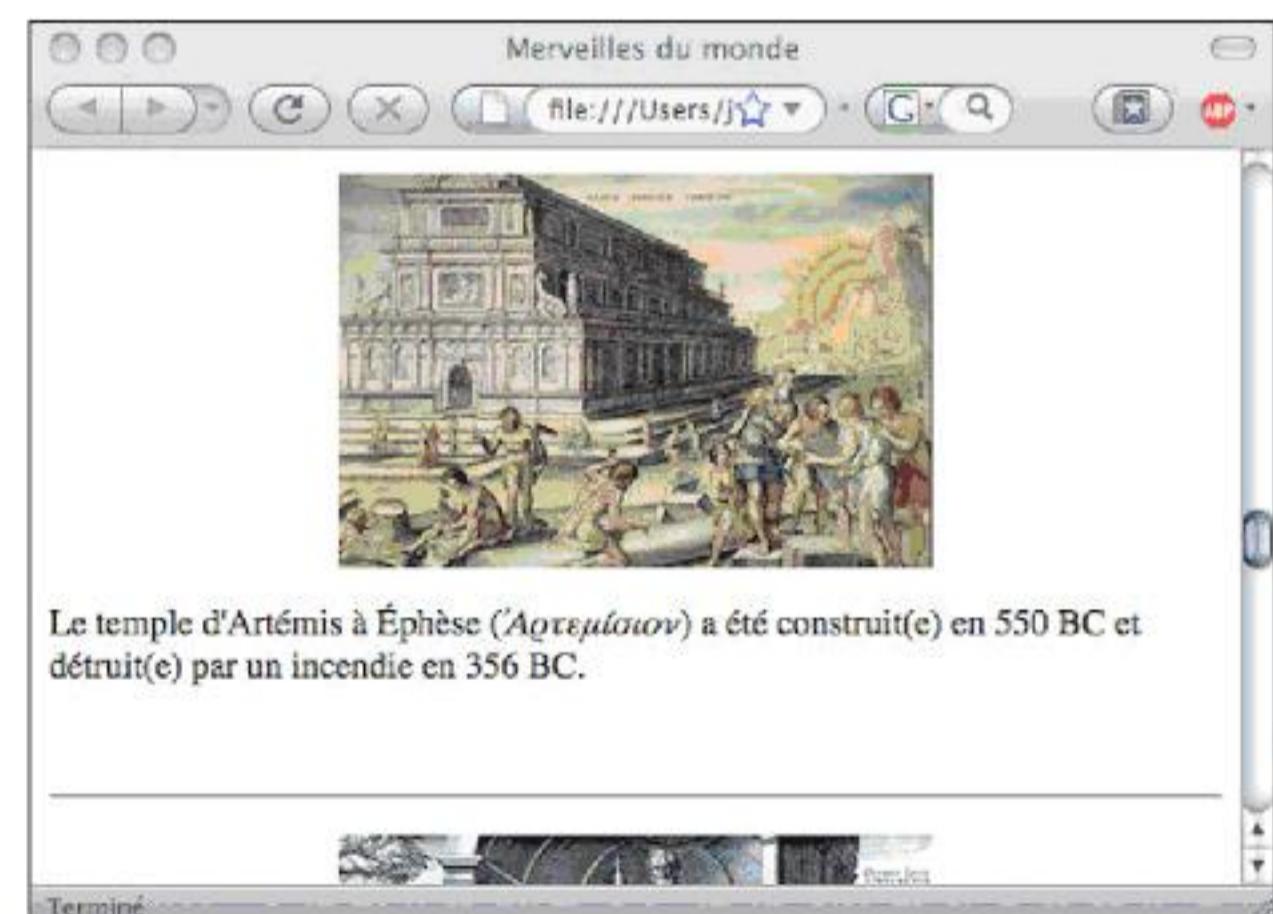
**Figure 4.15** Grâce à la fonction ceiling() de XPath, la largeur et la hauteur de l'image font désormais la moitié de leurs valeurs initiales.

## Arrondir les nombres

XPath dispose de trois fonctions pour arrondir les nombres : round() arrondit à l'entier le plus proche, ceiling(), au plus petit entier supérieur et floor(), au plus grand entier inférieur.

### CONSEIL

- Si vous utilisez la fonction format-number() vue dans la section précédente et que ce formatage provoque une perte de décimales, le processeur XSLT arrondit automatiquement le résultat du nombre produit.



**Figure 4.16** Les images des merveilles s'affichent en étant réduites de moitié.

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

```

x s l t

...
<xsl:template match="historique">
...
  <xsl:value-of select="histoire"/>
  <br /><br />
</xsl:template>
...

```

**Figure 4.23** Pour compléter la transformation du document XML, on ajoute l'élément *histoire* à la section "Historique".

Nom de la merveille	Ville	Pays	Âge	Hauteur
La grande pyramide de Khéops	Gizeh	Égypte	4578	138 mètres (452.6 pieds)
Le phare d'Alexandrie (ὁ Φάρος τῆς Ἀλεξανδρείας)	Alexandrie	Égypte	1602	117 mètres (383.8 pieds)
Le mausolée d'Halicarnasse (Μαυσωλεῖον Ἀλικαρνασσού)	Bodrum	Turquie	1755	41 mètres (134.5 pieds)
Le colosse de Rhodes (Κολοσσός της Ρόδου)	Rhodes	Grèce	56	32 mètres (105.0 pieds)
Le temple d'Artémis à Éphèse (Ἀρτεμίσιον)	Éphèse	Turquie	194	18 mètres (59.0 pieds)
La statue de Zeus à Olympie (Δίας μυθολογία)	Olympie	Grèce	855	12 mètres (39.4 pieds)
Les jardins suspendus de Babylone	Al Hillah	Irak	374	inconnue
Hauteur moyenne : 59.7 mètres				

## Historique



La grande pyramide de Khéops a été construite en 2570 BC et existe encore aujourd'hui. On pense qu'il a fallu 20 ans et 100000 ouvriers pour achever la Grande pyramide qui fut construite pour servir de tombe au pharaon Khéops de la Vème dynastie.

- normalize-space(*ch1*) renvoie *ch1* sans ses espaces de tête et de fin, avec toutes les suites d'espaces remplacées par une seule espace. Sans paramètre, normalize-space() s'applique au nœud courant.

## Fonctions booléennes

- not(*expression*) renvoie True si *expression* vaut False et False si *expression* vaut True.
- | (barre verticale). Bien que ce ne soit pas une fonction à proprement parler, elle permet de combiner deux ensembles de nœuds pour n'en former qu'un seul.

## CONSEIL

- Même si XPath 1.0 est encore la version la plus utilisée, elle ne fournit qu'un ensemble limité de fonctions. Si vous recherchez une fonctionnalité particulière, vérifiez qu'elle n'existe pas dans XPath 2.0, qui dispose d'un nombre de fonctions et d'opérateurs beaucoup plus grand. Sa spécification est disponible à partir de la page [www.w3.org/TR/xpath20/](http://www.w3.org/TR/xpath20/).

**Figure 3.24** Résultat final de toutes les transformations XSLT.

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

x s l - f o
<?xml version="1.0"?> <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">  <!-- structure générale --> <fo:layout-master-set> <fo:simple-page-master master-name="merveilles"> <fo:region-body/> </fo:simple-page-master> </fo:layout-master-set>  <!-- contenu de la page --> <fo:page-sequence master-reference="merveilles"> <fo:flow flow-name="xsl-region-body"> <fo:block>Le colosse de Rhodes</fo:block> </fo:flow> </fo:page-sequence> </fo:root>

**Figure 5.3** Document XSL-FO complet. Ce modèle de page s'appelle *merveilles* et contient un corps qui est rempli par le texte *Le colosse de Rhodes*. Vous remarquerez que, même si le contenu du document XSL-FO est très court, il doit quand même contenir les deux parties concernant la structure générale et le contenu de la page.

Le colosse de Rhodes

**Figure 5.4** Début du document PDF obtenu en traitant le document XSL-FO de la Figure 5.3.

## Créer un document XSL-FO

XSL-FO étant un langage de type XML, les documents XSL-FO respectent la syntaxe XML. La première ligne est donc la déclaration XML standard, et le document doit contenir un seul élément racine. Les documents XSL-FO sont des fichiers texte portant l'extension .fo.

La Figure 5.3 présente un document XSL-FO complet. On remarque que son élément racine est `<fo:root>` et qu'il déclare l'espace de noms XSL-FO (le préfixe `fo:`) avec l'attribut `xmlns:fo=http://www.w3.org/1999/XSL/Format`.

Son premier élément fils, `<fo:layout-master-set>`, débute la partie structure générale du document. Il contient l'élément `<fo:simple-page-master master-name="maître">`, où *maître* est le modèle de page qui sera utilisé plus loin. Il contient également un élément vide `<fo:region-body/>` indiquant que le contenu de cette page maître se trouvera dans le corps du résultat.

La partie contenu est comprise dans l'élément `<fo:page-sequence master-reference="maître">`, où *maître* est le nom qui a été défini dans la structure générale. Cet élément contient un élément `<fo:flow flow-name="xsl-region-body">` qui fait référence à la “région corps” définie dans la structure générale. C'est là que l'on utilise des éléments `<fo:block>` pour ajouter le contenu qui devra apparaître dans le résultat final.

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

```
x s l - f o

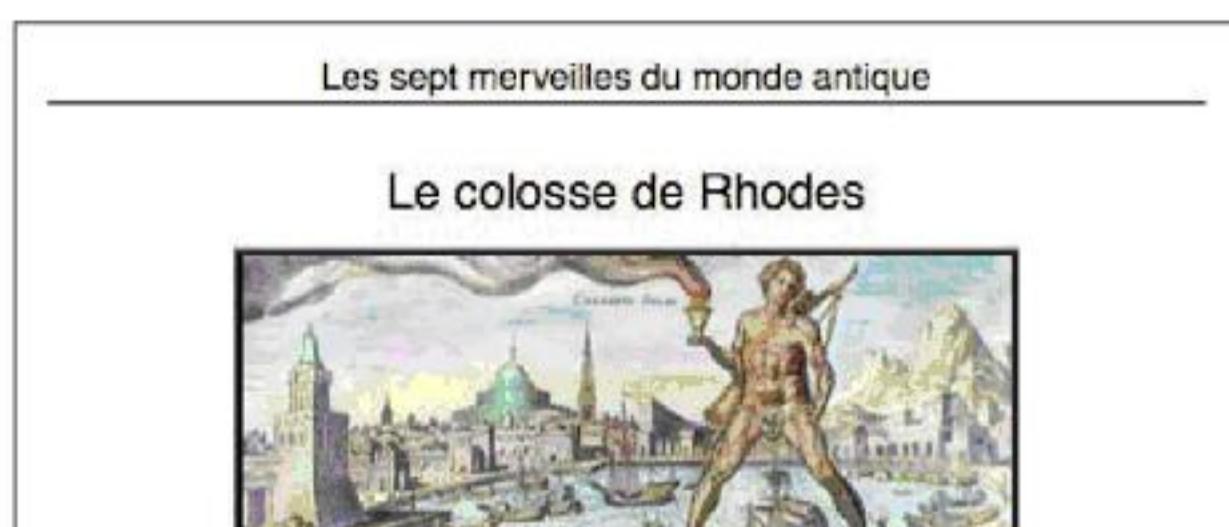
...
<!-- structure générale -->
<fo:layout-master-set>
  <fo:simple-page-master master-name="merveilles"
    page-width="21cm" page-height="11in">
    <fo:region-body margin="2.5cm" />
    <fo:region-before extent="2.5cm" />
  </fo:simple-page-master>
</fo:layout-master-set>
...
```

**Figure 5.11** L'en-tête (`fo:region-before`) aura 2,5 cm de haut. Pour éviter de recouvrir son contenu, le corps (`fo:region-body`) doit avoir également une marge de 2,5 cm. En outre, le corps ayant maintenant une marge de 2,5 cm, on supprime la configuration de la marge du modèle de page (`fo:simple-page-master`).

```
x s l - f o

...
<!-- contenu de la page -->
<fo:page-sequence master-
reference="merveilles">
  <fo:static-content flow-name="xsl-region-
before">
    <fo:block font-size="18pt"
      text-align="center"
      border-bottom-width="medium"
      border-bottom-style="solid"
      margin="0.6cm">
      Les sept merveilles du monde antique
    </fo:block>
  </fo:static-content>
...
```

**Figure 5.12** Le contenu de l'en-tête est défini dans l'élément `<fo:static-content flow-name="xsl-region-before">`, qui est un fils de l'élément `fo:page-sequence` dont l'attribut `master-reference` vaut "merveilles".



**Figure 5.13** Le contenu de l'en-tête, *Les sept merveilles du monde antique*, s'affiche sur 18 points, centré, avec un contour moyen, plein et aligné sur le bas.

## Créer un en-tête de modèle de page

Pour l'instant, nous n'avons utilisé que la région du modèle de page définie par l'élément `fo:region-body`, mais on peut également définir quatre autres régions correspondant respectivement à l'en-tête, au pied de page et aux encadrés gauche et droit.

La structure d'un en-tête de page est définie par l'élément `fo:region-before` (placé après l'élément `fo:region-body` à la Figure 5.11). Cet élément possède un attribut `extent="valeur"`, où `valeur` est la hauteur de l'en-tête. On peut également configurer d'autres attributs, comme `background` (arrière-plan) et `border` (bordure).

Pour définir le contenu de cet en-tête, on ajoute un élément `fo:static-content` à l'élément `fo:page-sequence` dont l'attribut `master-reference` est égal à la valeur de l'attribut `master-name` de `fo:simple-page-master` (voir Figure 5.12). L'attribut `flow-name` de ce `fo:static-content` doit être égal à "`xsl-region-before`" (qui désigne la région définie par l'élément `fo:region-before`).

Il reste ensuite à définir les éléments `fo:block` qui constitueront le contenu de l'en-tête.

### CONSEIL

- Bien qu'elles soient définies séparément, les régions de l'en-tête, du pied de page et des encadrés gauche et droit font partie du corps. Par conséquent, la marge du corps doit être égale (ou supérieure) à la valeur de l'attribut `extent` de la région de l'en-tête. Dans le cas contraire, le corps recouvrira l'en-tête.

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

```
x s l t
<!-- structure générale -->
<fo:layout-master-set>
  <fo:simple-page-master master-name=
    "couverture_merveilles"
    page-width="21cm" page-height="29.7cm"
    margin="2.5cm">
    <fo:region-body/>
  </fo:simple-page-master>
...

```

**Figure 5.24** Ce nouveau modèle de page est une page de présentation pour le document PDF et s'appelle couverture\_merveilles.

```
x s l t
...
<!-- page content -->
<fo:page-sequence master-reference="couverture_merveilles">
  <fo:flow flow-name="xsl-region-body">
    <fo:block font-size="28pt" text-align="center">
      Les sept merveilles du monde antique
    </fo:block>
...

```

**Figure 5.25** Une version initiale de la page de couverture apparaissait dans les Figures 5.1 et 5.2.



**Figure 5.26** La feuille de style utilise un modèle de page différent, sans en-tête ni pied de page.

## Ajouter un nouveau modèle de page

Dans la section “Définir un modèle de page”, nous avons défini un modèle de page que nous avons ensuite utilisé pour produire les résultats de tous nos exemples. Nous voulons maintenant créer une page de couverture ayant une structure générale et un contenu différents.

Pour cela, nous devons ajouter un nouveau modèle de page en ajoutant un élément fils `fo:simple-page-master` à `fo:layout-master-set` (voir Figure 5.24). Ce nouvel élément aura un attribut `master-name="maître"`, où `maître` est le nom du modèle de page utilisé dans l’élément `fo:page-sequence`, et contiendra un élément `<fo:region-body/>` indiquant que le contenu de cette page ira dans son corps. Comme on l’a vu dans la section “Créer un en-tête de modèle de page”, on peut également lui ajouter d’autres éléments de déclaration de régions.

Pour définir le contenu de page de ce nouveau modèle, on ajoute un élément `<fo:page-sequence master-reference="maître">`, où `maître` est le nom du modèle que nous venons de définir (voir Figure 5.25). Celui-ci contiendra un élément `<fo:flow flow-name="xsl-region-body">` permettant de créer le conteneur des blocs de contenu du corps. Si l’on a défini d’autres régions, on répétera cette dernière étape.

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

# 6

## Création d'une DTD

Dans la première partie de ce livre, vous avez appris la grammaire sous-jacente de XML, qui regroupe les règles d'écriture d'un document XML. Dans la deuxième partie, vous avez vu comment transformer un document XML en HTML. Vous allez maintenant apprendre à définir un langage à balises personnalisé en XML.

Pour ce faire, vous devez d'abord identifier ses éléments et leurs attributs, indiquer ceux qui sont obligatoires et ceux qui ne le sont pas.

Ces informations forment ce que l'on appelle un *schéma*. Un historien, par exemple, pourrait créer le langage MdmML (*Merveilles du monde Markup Language*) pour cataloguer les données sur les merveilles du monde. Les éléments de MdmML pourraient être `merveille`, `nom`, `année_construction`, `histoire`, etc.

Les schémas, bien que non obligatoires, ont une importance toute particulière pour la cohérence des documents XML. En fait, vous pouvez confronter n'importe quel document XML à son schéma afin de vérifier s'il respecte les règles de celui-ci. Si c'est le cas, on dit alors que le document est *valide* et on peut être sûr que ses données sont sous la forme exigée par le schéma.

Il existe essentiellement deux systèmes d'écriture des schémas : les DTD et XML Schema. Les DTD, ou *définition de type de document*, sont l'ancien système – toujours largement utilisé – avec une syntaxe particulière et relativement limitée. Les trois chapitres qui suivent leur sont consacrés. L'autre système, XML Schema, est décrit en détail dans la quatrième partie de cet ouvrage. Pour connaître les raisons qui pourraient vous pousser à préférer l'un de ces systèmes à l'autre, consultez la section “Avantages et inconvénients des DTD”, page 108.

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

x m l
<pre>&lt;?xml version="1.0"?&gt; &lt;anciennes_merveilles&gt;   &lt;merveille&gt;     &lt;nom langue="Français"&gt;Le colosse de Rhodes&lt;/nom&gt;     &lt;nom langue="Grec"&gt;Κολοσσός της Ρόδου&lt;/nom&gt;     &lt;lieu&gt;Rhodes en Grèce&lt;/lieu&gt;     &lt;hauteur unité="mètres"&gt;32&lt;/hauteur&gt;     &lt;historique&gt;       &lt;année_construction ère="BC"&gt;282       &lt;/année_construction&gt;       &lt;année_destruction ère="BC"&gt;226       &lt;/année_destruction&gt;       &lt;motif_destruction&gt;tremblement de terre&lt;/motif_destruction&gt;       &lt;histoire&gt;En 294 avant J. -C., les habitants de l'île de Rhodes...&lt;/histoire&gt;     &lt;/historique&gt;     &lt;image_principale fichier="lighthouse.jpg" l="528" h="349"/&gt;     &lt;source idsection="112" idjournal="53"/&gt;   &lt;/merveille&gt; &lt;/anciennes_merveilles&gt;</pre>

**Figure 6.7** Dans cette version allégée de notre document XML, l'élément `anciennes_merveilles` a un seul élément fils, l'élément `merveille`.

## Définir un élément contenant un fils

Maintenant que vous avez compris comment définir les éléments XML de base dans une DTD, il faut que vous puissiez faire de même pour les éléments pères, c'est-à-dire ceux qui contiennent d'autres éléments.

Cette définition est de la forme `<!ELEMENT balise (fils)>` où `balise` est le nom de l'élément père et `fils`, le nom de l'élément qui sera contenu dans `balise`.

### CONSEILS

- Une balise définie pour avoir un élément fils ne peut rien contenir hormis cet élément. Elle ne peut notamment pas inclure d'autres éléments ni du texte.
- L'élément fils peut être défini comme étant facultatif ou comme pouvant apparaître plusieurs fois. Pour plus de détails, voir la section "Définir plusieurs occurrences", plus bas dans ce chapitre.
- Comme nous le verrons à la page suivante, vous pouvez également contrôler l'ordre d'apparition des éléments dans un document XML.
- Un élément qui a été défini comme devant contenir un autre élément doit inclure celui-ci à chaque fois. Dans le cas contraire, le document XML n'est pas considéré comme valide.

d t d
<pre>&lt;!ELEMENT anciennes_merveilles (merveille)&gt;</pre>

**Figure 6.8** Cette DTD précise que l'élément `anciennes_merveilles` ne peut contenir qu'un seul élément `merveille`, comme à la Figure 6.7. Le contenu de l'élément `merveille` ne dépend que de sa définition (il n'est pas concerné par la définition de `anciennes_merveilles`, par exemple).

*image  
not  
available*

*image  
not  
available*

*image  
not  
available*

```

x m l

<anciennes_merveilles>
  <merveille>
    <nom>Le colosse de Rhodes</nom>
    <lieu>Rhodes en Grèce</lieu>
  </merveille>
  <merveille>
    La grande pyramide de Kheops, Gizeh
    en Égypte
  </merveille>
  <merveille>
    Le temple d'Artémis à Éphèse
    <ville>Éphèse</ville>
    <pays>Turquie</pays>
  </merveille>
  <merveille>
    <nom>Le mausolée d'Halicarnasse</nom>
    <lieu>
      <ville>Bodrum</ville>
      <pays>Turquie</pays>
    </lieu>
  </merveille>
</anciennes_merveilles>

```

**Figure 6.16** On a ajouté ici un nouvel élément *merveille* (provenant d'une source inattendue) au document XML de la Figure 6.13. Cet élément a une nouvelle structure : son élément *lieu* ne contient aucune donnée PCDATA mais a deux éléments fils, *ville* et *pays*.

```

x m l

<!ELEMENT anciennes_merveilles (merveille+)>
<!ELEMENT merveille (#PCDATA | nom | lieu
                      | ville | pays)*>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT lieu ANY>
<!ELEMENT ville (#PCDATA)>
<!ELEMENT pays (#PCDATA)>

```

**Figure 6.17** Au lieu d'ajouter la nouvelle structure *lieu* à la DTD de la Figure 6.14, on définit cet élément pour qu'il puisse contenir n'importe quoi. Ce n'est pas aussi clair que définir *lieu* avec un contenu spécifique, mais cela fonctionne.

## Définir un élément pouvant contenir n'importe quoi

Bien que cela ne soit pas idéal pour créer un ensemble structuré de règles, une DTD peut définir un élément pouvant contenir n'importe quoi, c'est-à-dire n'importe quelle combinaison d'éléments et de texte. Comme pour les contenus mixtes, cette possibilité sert surtout à créer une DTD permettant de gérer des documents XML provenant de plusieurs sources différentes. C'est parfois le seul moyen de définir des éléments connus tout en autorisant des structures d'éléments inconnues (voir Figure 6.16).

La syntaxe de ce type d'élément est `<!ELEMENT balise ANY>`, où *balise* est le nom de l'élément qui pourra contenir n'importe quoi.

### CONSEILS

- Utilisez ANY avec parcimonie car le but d'une DTD est de définir des règles sur le contenu exact d'un élément. Si vous autorisez tous les éléments à contenir n'importe quoi, autant vous passer d'une DTD : les DTD ne sont pas obligatoires, elles vous aident simplement à maintenir la cohérence des données.
- ANY indique qu'un élément peut contenir n'importe quelle structure. Cependant, s'il contient des éléments fils, ceux-ci doivent quand même être définis dans la DTD. En d'autres termes, ANY n'autorise pas un élément à contenir des éléments fils qui n'apparaissent pas dans la DTD car tous les éléments d'un document XML valide doivent être définis.
- Tout élément ne doit être défini qu'une seule fois dans une DTD, même s'il peut apparaître à différents endroits d'un document XML valide.

*image  
not  
available*

*image  
not  
available*

## Définir des valeurs par défaut

Au lieu d'utiliser #REQUIRED ou #IMPLIED, vous pouvez donner une valeur par défaut à un attribut. Pour cela, vous pouvez faire suivre le type de l'attribut d'une valeur entre guillemets, comme à la Figure 6.23. En ce cas, l'attribut recevra cette valeur si le document XML ne lui en donne pas. Vous pouvez également utiliser la syntaxe #FIXED "d*efault*" : en ce cas, l'attribut recevra la valeur "d*efault*" si le document ne lui en donne pas et, si cet attribut est initialisé dans le document, il devra l'être avec cette valeur, comme à la Figure 6.25.

### CONSEILS

- Si vous définissez un attribut avec une valeur par défaut, le processeur XML ajoutera automatiquement cette valeur à l'attribut si le document ne l'a pas initialisé (voir Figure 6.24).
- Si l'attribut est défini avec #FIXED "d*efault*", la valeur de l'attribut dans le document XML devra être "d*efault*". S'il n'est pas initialisé, le processeur XML lui ajoutera automatiquement cette valeur (voir Figure 6.26).
- Vous pouvez combiner une valeur par défaut et #REQUIRED ou #IMPLIED. Cela dit, comme une valeur par défaut a déjà été fixée, aucun de ces états n'a vraiment de sens.

d t d
<!ELEMENT hauteur (#PCDATA)>
<!ATTLIST hauteur unité CDATA "mètres">

**Figure 6.23** Ajout de la valeur "mètres" par défaut à l'attribut hauteur.

x m 1
<hauteur unité="pieds">39</hauteur>
<hauteur unité="mètres">39</hauteur>
<hauteur>39</hauteur>

**Figure 6.24** Tous ces extraits XML sont valides. L'attribut unité peut recevoir n'importe quelle valeur, voire être absent : en ce cas, comme dans le troisième exemple, le processeur agira comme si l'attribut était présent et qu'il valait "mètres".

d t d
<!ELEMENT hauteur (#PCDATA)>
<!ATTLIST hauteur unité CDATA #FIXED "mètres">

**Figure 6.25** Une valeur fixée permet de garantir qu'un attribut aura toujours une certaine valeur, qu'il apparaisse ou non dans le document XML.

x m 1
<hauteur unité="mètres">39</hauteur>
<hauteur unité="pieds">39</hauteur>
<hauteur>39</hauteur>

**Figure 6.26** Ces exemples sont identiques à ceux de la Figure 6.24. Par rapport à la DTD de la Figure 6.25, l'exemple du milieu n'est plus valide : si l'attribut est initialisé, il doit contenir la valeur "mètres" (ni "pieds" ni une autre valeur). Dans l'exemple du bas, le processeur agit comme si l'attribut valait "mètres".

d t d
<!ELEMENT hauteur (#PCDATA)> <!ATTLIST hauteur unité (mètres pieds) #REQUIRED>

**Figure 6.27** L'attribut *unité* de l'élément *hauteur* ne pourra prendre que deux valeurs possibles : *mètres* ou *pieds*. Notez que l'attribut devra également être initialisé (à cause de l'option *#REQUIRED*).

x m l
<hauteur unité="mètres">39</hauteur>
<hauteur unité="pouces">39</hauteur>
<hauteur>39</hauteur>

**Figure 6.28** Parmi ces trois extraits XML, seul le premier est valide par rapport à la DTD de la Figure 6.27. L'exemple du milieu n'est pas valide parce que "pouces" ne fait pas partie des choix autorisés pour le contenu de l'attribut. L'exemple du bas ne l'est pas non plus parce que l'attribut *unité* n'a pas été fourni alors qu'il a été déclaré avec *#REQUIRED*.

## Définir des attributs avec des choix

Le type des attributs n'est pas limité à CDATA. Vous pouvez en effet préciser qu'un attribut pourra recevoir différentes valeurs prédéfinies, en plaçant ces valeurs entre parenthèses et en les séparant par une barre droite, comme à la Figure 6.27.

### CONSEILS

- Chaque choix de la liste doit respecter les règles des noms XML.
- Il existe également les types d'attributs *ID*, *IDREF* et *IDREFS*, qui seront expliqués dans les deux sections qui suivent ; *NMTOKEN* et *NMTOKENS*, qui seront présentés dans la section "Limiter les attributs à des noms XML valides" et *ENTITY*, qui sera décrit au Chapitre 7.

## Définir des attributs ayant des valeurs uniques

Certains types d'attributs sont spéciaux. Les attributs définis de type ID ont une valeur unique pour tout le document XML. Un attribut ID est donc idéal pour représenter des clés ou autres informations d'identification (codes de produits, identifiants de clients, etc.). Ce type d'attribut est nécessairement associé à l'option #REQUIRED ou #IMPLIED ; il ne peut pas utiliser de valeurs par défaut.

### CONSEILS

- *Un document XML ayant deux éléments avec des attributs ID de même valeur n'est pas valide, que les noms des éléments soient les mêmes ou non.*
- *La seule exception à cette règle est qu'il peut y avoir un nombre quelconque d'attributs ID omis dans le document, qui auront tous une valeur nulle par défaut.*
- *La valeur d'un attribut ID doit respecter les règles des noms XML : cela signifie qu'un attribut ID ne peut pas contenir une chaîne uniquement numérique, comme c'est le cas de nombreux champs d'identifiants dans les bases de données, le numéro INSEE, etc., sauf si vous les préfixez avec une lettre ou un blanc souligné.*

d t d
<!ELEMENT merveille (nom)>
<!ATTLIST merveille code ID #REQUIRED>

**Figure 6.29** Il est préférable de rendre obligatoire un attribut ID ; dans le cas contraire, il doit être implicite puisque ce type d'attribut ne peut pas avoir de valeur par défaut.

x m l
<merveille code="w_143"> <nom langue="Français">Les jardins suspendus de Babylone</nom> </merveille> <merveille code="w_284"> <nom langue="Français">La statue de Zeus à Olympie</nom> </merveille>

x m l
<merveille code="w_284"> <nom langue="Français">Les jardins suspendus de Babylone</nom> </merveille> <merveille code="w_284"> <nom langue="Français">La statue de Zeus à Olympie</nom> </merveille>

**Figure 6.30** D'après la DTD de la Figure 6.29, l'attribut code doit avoir une valeur unique dans tout le document XML. Le premier extrait est donc valide, mais pas le second.

```
d t d
<!ELEMENT site_spécialisé (titre, url)>
<!ATTLIST site_spécialisé
    merveille_concernée IDREF #REQUIRED>
```

**Figure 6.31** L'élément *site\_spécialisé* mémorise les sites web consacrés à chaque *merveille*. Son attribut *merveille\_concernée* est de type *IDREF* afin de pouvoir contenir l'*ID* de la *merveille* auquel il s'intéresse.

```
x m l
<site_spécialisé merveille_concernée="w_143">
    <titre>Les jardins perdus</titre>
    <url>www.jardins-perdus.com</url>
</site_spécialisé>
<site_spécialisé merveille_concernée="w_143">
    <titre>Hérodote à Babylone</titre>
    <url>www.herodotes.com/babylone</url>
</site_spécialisé>
<site_spécialisé merveille_concernée="w_284">
    <titre>Zeus à Olympie</titre>
    <url>www.olympiezeus.com</url>
</site_spécialisé>
```

**Figure 6.32** D'après la DTD de la Figure 6.31, l'attribut *merveille\_concernée* doit contenir une valeur correspondant à un attribut *ID* du document (cet extrait provient d'un document XML qui contient également le premier exemple de la Figure 6.30 de la page précédente). Attention : ces sites sont imaginaires...

```
d t d
<!ELEMENT site_généraliste (titre, url)>
<!ATTLIST site_généraliste
    contenu IDREFS #REQUIRED>
```

**Figure 6.33** L'élément *site\_généraliste* a un attribut *IDREFS* nommé *contenu* qui contient une liste des *ID* des *merveilles* auxquels il s'intéresse.

```
x m l
<site_généraliste contenu="w_143 w_284">
    <titre>Merveilles du monde</titre>
    <url>http://www.merveilles_du_monde.com/
    </url>
</site_généraliste>
```

**Figure 6.34** Cet extrait XML est valide par rapport à la DTD de la Figure 6.33.

## Utiliser les attributs ayant des valeurs uniques

Un attribut *IDREF* est un attribut dont la valeur est celle d'un attribut *ID* existant dans le document XML (voir Figure 6.31).

Un attribut *IDREFS* est un attribut dont la valeur est une liste de valeurs d'attributs *ID* existants séparées par des espaces (voir Figure 6.33).

### CONSEILS

- Comme le montre la Figure 6.32, plusieurs attributs *IDREF* peuvent référencer le même *ID*. C'est uniquement l'*ID* lui-même qui doit être unique à un élément.
- Rien n'empêche un attribut *IDREFS* de contenir plusieurs fois la même valeur : *contenu="w\_143 w\_143 w\_143"*, par exemple, est tout à fait valide, même si ce n'est pas ce que vous souhaitez. Pour mieux contrôler les éléments et le contenu de leurs attributs, vous devez abandonner les *DTD* et vous tourner vers *XML Schema* (voir Partie 4).

## Limiter les attributs à des noms XML valides

Bien que la syntaxe des DTD soit assez limitée, vous pouvez quand même appliquer une restriction aux attributs : la valeur d'un attribut de type NMTOKEN doit être un nom XML valide, c'est-à-dire une valeur commençant par une lettre ou un blanc souligné, suivis uniquement de lettres, de chiffres, de blancs soulignés, de tirets et de points.

Si vous voulez que la valeur de l'attribut soit une liste de noms XML valides, séparés par des espaces, utilisez plutôt le type NMOKENS.

Ces deux types peuvent être accompagnés des options #REQUIRED ou #IMPLIED.

### CONSEILS

- *Les attributs NMTOKEN ne peuvent pas contenir d'espaces, ce qui peut être l'une des raisons pour lesquelles on souhaite utiliser ce type particulier.*
- *Si vous voulez que la valeur d'un attribut soit non seulement un nom XML valide mais également unique dans tout le document XML, utilisez ID au lieu de NMTOKEN.*

d t d
<!ELEMENT visite EMPTY> <!ATTLIST visite mot_clé NMTOKEN #REQUIRED>

**Figure 6.35** Dans cet exemple un peu tiré par les cheveux, on a besoin pour chaque merveille d'un mot-clé qui servira de clé primaire dans l'application Visite des merveilles. Pour que ce mot-clé soit limité à un seul mot, sans espace, on utilise un attribut NMTOKEN.

x m 1
<merveille> <visite mot_clé="colosse"/> <nom langue="Français">Le colosse de Rhodes </nom> ... <merveille> <visite mot_clé="grande pyramide"/> <nom langue="Français">La grande pyramide de Kheops</nom> ...

x m 1
<merveille> <visite mot_clé="colosse"/> <nom langue="Français">Le colosse de Rhodes </nom> ... <merveille> <visite mot_clé="grande_pyramide"/> <nom langue="Français">La grande pyramide de Kheops</nom> ...

**Figure 6.36** Seul le second exemple est valide par rapport à la DTD de la Figure 6.35. Dans le premier, la valeur "grande pyramide" de l'attribut mot\_clé comprend une espace, ce qui est interdit pour les attributs NMTOKEN.

# 7

## Entités et notations dans les DTD

Les entités sont simplement des entrées textuelles automatiques, c'est-à-dire des raccourcis d'écriture. Grâce à elles, vous pouvez associer un nom à un texte qui sera substitué à l'entité à chaque fois qu'elle apparaît dans le document.

Il existe plusieurs types d'entités qui fonctionnent toutes de la même façon et qui sont toutes définies au sein d'une DTD. Leurs différences portent sur les endroits où la substitution pourra avoir lieu et sur le type de données qu'elles peuvent contenir.

Ces types peuvent être classés en deux catégories : les *entités générales* et les *entités paramètres*. Les premières ne peuvent être substituées que dans un document XML, tandis que les secondes ne peuvent l'être que dans une DTD.

Les entités générales peuvent elles-mêmes être subdivisées en entités internes et externes, analysées ou non analysées. Les entités paramètres peuvent aussi être subdivisées en entités internes et externes, mais elles sont toujours analysées.

## Créer une entité générale

La forme d’entité la plus simple est définie dans une DTD et représente simplement du texte. Officiellement, on les appelle *entités générales internes*, mais on les désigne souvent sous le terme de raccourcis.

Ces entités sont créées à l’aide de la syntaxe `<!ENTITY nom_ent contenu>`, où *nom\_ent* est le nom que l’on souhaite donner à l’entité et qui sera celui sous lequel elle sera désignée dans le document XML ; *contenu* est le texte qui sera substitué à l’entité dans le document.

### CONSEILS

- *Le nom de l’entité doit respecter les règles des noms XML valides.*
- *En XML, il existe déjà cinq entités générales prédéfinies : &amp;, &lt;, &gt;, &quot; et &apost; (voir page 14). Toutes les autres entités doivent être déclarées dans la DTD avant leur utilisation.*
- *De nombreuses entités courantes ont déjà été définies, prêtées à être incluses dans vos propres DTD. Pour plus de détails, voir les conseils de la page 95.*

d t d
<!ENTITY mdm "merveilles du monde">

**Figure 7.1** Les entités générales internes facilitent l’inclusion de phrases longues qui reviennent souvent dans un document XML.

```
x m l
<histoire>
Le fait le plus intéressant à propos des
jardins est qu'il existe une forte controverse
sur leur existence même
...
Dans tous les cas, il est intéressant de noter
que l'imagination des poètes et des anciens
historiens a créé l'une des sept &mdash;.
</histoire>
```

**Figure 7.2** Il est plus simple et plus rapide de taper &mdash;, que merveilles du monde.

```
h t m l
<p align="center">
<strong>LES JARDINS SUSPENDUS DE BABYLONE
</strong>
<br></p>
Les jardins suspendus de Babylone a été
construit en 600 BC
et détruit(e) par un tremblement de terre en
226 BC.
...
Le fait le plus intéressant à propos des jardins
est qu'il existe une forte controverse sur leur
existence même
...
Dans tous les cas, il est intéressant de noter
que l'imagination des poètes et des anciens
historiens
a créé l'une des sept merveilles du monde.
<br><br>
...
```

**Figure 7.3** L'entité de la Figure 7.2 est substituée lors de l'analyse du document XML.

## Utiliser les entités générales

Pour utiliser dans un document XML une entité que vous avez définie dans une DTD, il suffit de préfixer son nom par une esperluette (&) et de le faire suivre d'un point-virgule (;) : l'entité *no\_ent*, par exemple, s'utilisera donc sous la forme &*no\_ent*;.

### CONSEILS

- Vous ne pouvez utiliser une entité générale que si elle a été définie dans la DTD utilisée par votre document XML. Dans le cas contraire, le processeur affirmera qu'elle n'existe pas.
- Les références de caractères permettent d'ajouter des symboles spéciaux : &#246;, par exemple, produit le caractère ö. Bien qu'elles leur ressemblent, ce ne sont pas des entités et elles n'ont pas besoin d'être déclarées dans la DTD (voir Annexe B).
- Les entités générales ne peuvent être utilisées que par les documents XML, pas par les feuilles XSLT (il existe des moyens de contourner cette restriction, mais ils sont complexes et lourds).
- Vous pouvez utiliser une entité dans une autre définition d'entité du moment que cela n'induit pas de référence cyclique.

## Créer une entité générale externe

Si le texte de votre entité est plus long ou si elle doit pouvoir être utilisée dans plusieurs DTD, il est souvent plus pratique de la sauvegarder dans un fichier séparé portant l'extension .ent.

Quand vous aurez besoin de cette entité dans une DTD, il suffira d'utiliser la syntaxe `<!ENTITY nom_ent SYSTEM entite.uri>`, où *nom\_ent* est le nom de l'entité, **SYSTEM** indique qu'il s'agit d'une entité externe et *entite.uri* est l'emplacement du fichier comprenant le contenu de cette entité.

### CONSEILS

- *entite.uri* peut désigner un fichier local à votre ordinateur ou un fichier situé sur une machine du réseau local ou sur Internet.
- L'extension .ent n'est pas obligatoire (n'importe quelle autre extension convient), mais c'est la plus employée.
- Grâce aux entités externes, vous pouvez en fait créer une DTD à partir de plusieurs autres.

d t d
<histoire>
Le fait le plus intéressant à propos des jardins est qu'il existe une forte controverse sur leur existence même.
...
Dans tous les cas, il est intéressant de noter que l'imagination des poètes et des anciens historiens a créé l'une des sept &mdash;.
</histoire>

**Figure 7.4** Ici, on définit l'entité pour qu'elle représente une portion de texte XML complète. Vous remarquerez qu'elle inclut l'entité mdm définie à la Figure 7.1. Ce texte est sauvegardé dans le fichier jardins.ent.

d t d
<!ENTITY histoire_jardins SYSTEM "jardins.ent">

**Figure 7.5** L'entité histoire\_jardins pointe vers l'URI du fichier contenant le texte de l'entité.

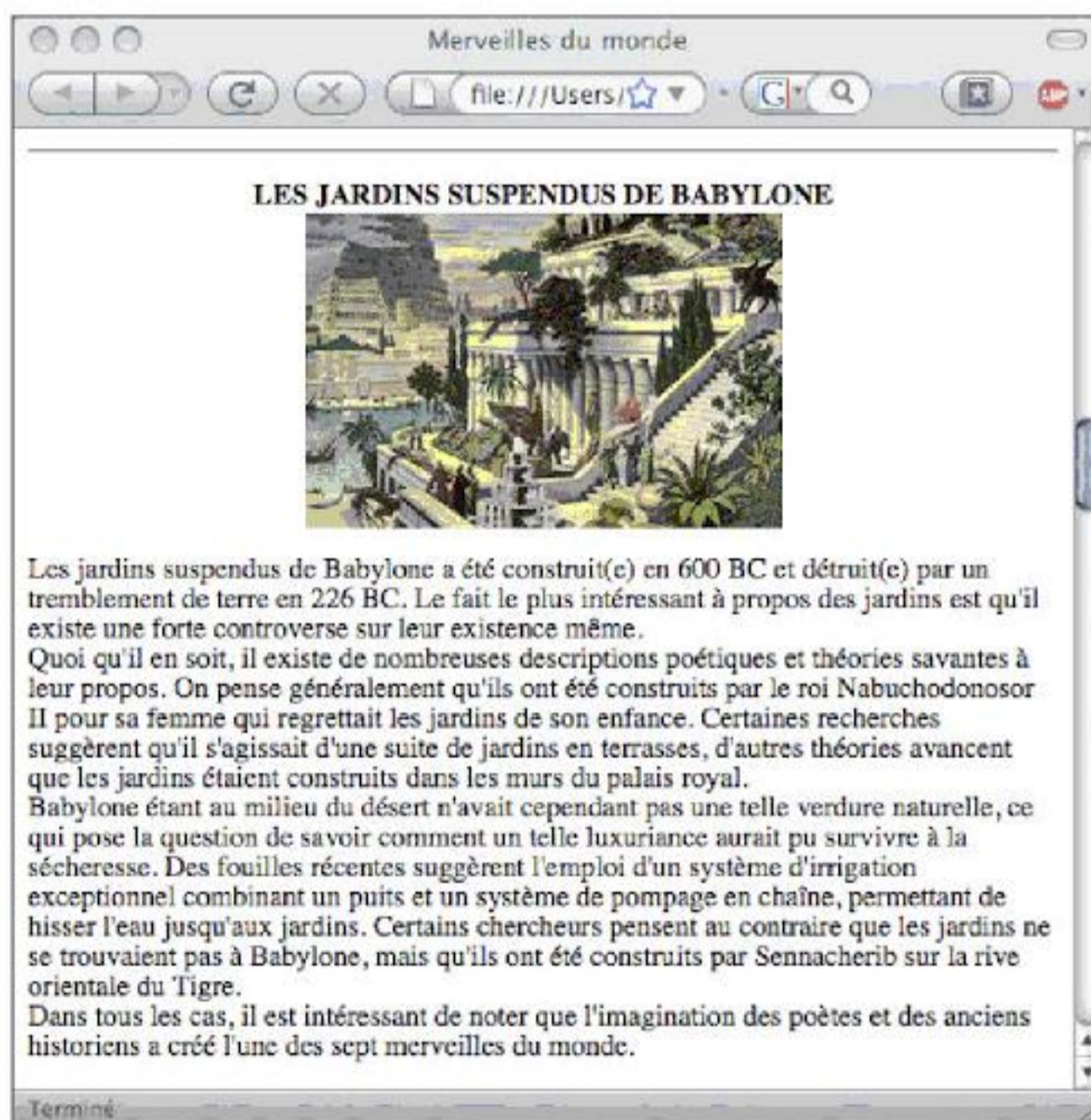
```

x m 1

<?xml version="1.0" standalone="no"?>
...
<merveille>
  <nom langue="Français">Les jardins
  suspendus de Babylone</nom>
  <lieu>Al Hillah en Irak</lieu>
  <hauteur unité="mètres">0</hauteur>
  <historique>
    <année_construction ère="BC">600</année_
    construction>
    <année_destruction ère="BC">226</année_
    destruction>
    <motif_destruction>tremblement de terre
    </motif_destruction>
    &histoire_jardins;
  </historique>
...

```

**Figure 7.6** Assurez-vous de placer `standalone="no"` dans la déclaration XML. Vous pouvez ensuite utiliser l'entité générale externe `&histoire_jardins;` dans le document.



**Figure 7.7** L'entité définie à la Figure 7.5 contient un élément, du texte et une autre entité. Le principe est le même : on tape un texte court et le processeur le remplace par le contenu qu'il désigne. Notez que tous les éléments provenant d'une entité générale externe doivent quand même être définis dans la DTD pour que le document soit valide.

## Utiliser les entités générales externes

Une entité externe peut être utilisée et partagée avec d'autres utilisateurs. Vous pouvez également emprunter leurs entités externes.

Pour utiliser une entité générale externe, vous devez ajouter `standalone="no"` à la déclaration XML, comme à la Figure 7.6. Cette directive indique au processeur que le document a besoin d'un fichier externe ; ici, celui qui contient la définition de l'entité. Ensuite, l'entité s'utilise exactement comme une entité interne.

### CONSEILS

- Une *URI* (Uniform Resource Identifier) est une chaîne de caractères permettant d'identifier et de localiser une ressource. On la confond souvent avec une *URL* (Uniform Resource Locator) bien que, techniquement, une *URI* puisse être une *URL* ou un *URN* (Universal Resource Name). Dans le cadre de ce livre, *URI* et *URL* ont la même signification.
- Vous pouvez utiliser un lien vers une liste d'entités standard, comme [www.w3.org/TR/xhtml1/#h-A2](http://www.w3.org/TR/xhtml1/#h-A2). Cela vous permet d'utiliser des références d'entités mnémotechniques pour les caractères accentués sans devoir les définir vous-même.
- Une entité générale (comme celles que nous avons décrites dans les pages précédentes) est définie comme une composante de la DTD et est utilisée dans un document XML. Il existe un autre type d'entité qui permet d'ajouter du contenu à la DTD elle-même. Ces entités sont appelées entités paramètres et utilisent une syntaxe un peu différente (voir la section "Créer et utiliser des entités paramètres", plus loin dans ce chapitre).

## Créer des entités pour du contenu non analysé

Pour l'instant, nous n'avons présenté que des entités représentant du texte. Celles-ci sont également appelées *entités analysées* car le processeur les lit et les analyse pendant le parcours du document XML. Les *entités non analysées*, que nous allons étudier ici, ne représentent généralement pas du texte (bien qu'elles le puissent) et sont totalement ignorées du processeur. Elles peuvent donc servir à intégrer du contenu non textuel ou non XML dans un document XML. Il peut s'agir de n'importe quoi : du texte, un fichier image ou vidéo, un document PDF, etc.

Pour créer une entité représentant ce type de contenu, il faut d'abord indiquer dans une DTD comment le traiter à l'aide d'une *notation*. La syntaxe d'une notation est de la forme `<!NOTATION nom_n SYSTEM instr_n>`, où *nom\_n* est le nom permettant d'identifier le contenu non analysé, *instr\_n* est généralement une information (comme une URI) qui définit le traitement du contenu (il n'existe pas de format officiel et vous devez donc consulter la documentation de votre processeur XML pour plus de détails).

Pour définir une entité désignant un contenu non analysé, on utilise la syntaxe `<!ENTITY nom_ent SYSTEM entité.uri NDATA nom_n>`, où *entité.uri* désigne l'emplacement du fichier du contenu non analysé et *nom\_n* est le nom de la notation créée précédemment.

### CONSEILS

- Lors de la création de la notation, *instr\_notation* peut être un type MIME (qui est un standard Internet permettant de décrire des types de contenus), une URI pointant vers une application interne ou externe capable de traiter le contenu non analysé, ou quasiment n'importe quoi (cela n'a aucune importance puisque le processeur XML ne le traite pas). Selon la spécification du W3C, il n'existe pas de format standard pour cette information et chaque application XML peut l'utiliser comme elle le souhaite.
- Les entités non analysées sont nécessairement des entités générales car elles sont intégrées au corps d'un document XML.



**Figure 7.8** Une donnée non analysée classique : une image JPEG, stockée dans le fichier `lighthouse.jpg`.

d t d
<pre> &lt;!ELEMENT anciennes_merveilles (merveille*)&gt; &lt;!ELEMENT merveille (nom+, photo)&gt; &lt;!ELEMENT nom (#PCDATA) &gt; &lt;!ATTLIST nom     langue CDATA #REQUIRED&gt;  &lt;!NOTATION jpg SYSTEM "image/jpeg"&gt; </pre>

**Figure 7.9** Le nom identifiant l'élément de notation jpg sera utilisé lors de la création de l'entité pour le contenu non analysé.

d t d
<pre> &lt;!ELEMENT anciennes_merveilles (merveille*)&gt; &lt;!ELEMENT merveille (nom+, photo)&gt; &lt;!ELEMENT nom (#PCDATA) &gt; &lt;!ATTLIST nom     langue CDATA #REQUIRED&gt;  &lt;!NOTATION jpg SYSTEM "image/jpeg"&gt;  &lt;!ENTITY image_phare SYSTEM     "lighthouse.jpg" NDATA jpg&gt; </pre>

**Figure 7.10** Le nom de l'entité, `image_phare`, fait référence à un fichier externe, appelé `lighthouse.jpg`. On peut obtenir plus d'informations sur ce fichier en examinant l'identifiant de notation jpg.

```

d t d

<!ELEMENT anciennes_merveilles (merveille*)>
<!ELEMENT merveille (nom+, photo)>
<!ELEMENT nom (#PCDATA) >
<!ATTLIST nom
    langue CDATA #REQUIRED>

<!NOTATION jpg SYSTEM "image/jpeg">

<!ENTITY image_phare SYSTEM
    "lighthouse.jpg" NDATA jpg>
<!ELEMENT photo EMPTY>
<!ATTLIST photo
    source ENTITY #REQUIRED>

```

**Figure 7.11** On définit d'abord l'élément `photo` qui contiendra l'élément `source` désignant les données non analysées. Puis on définit cet attribut comme étant de type `ENTITY`.

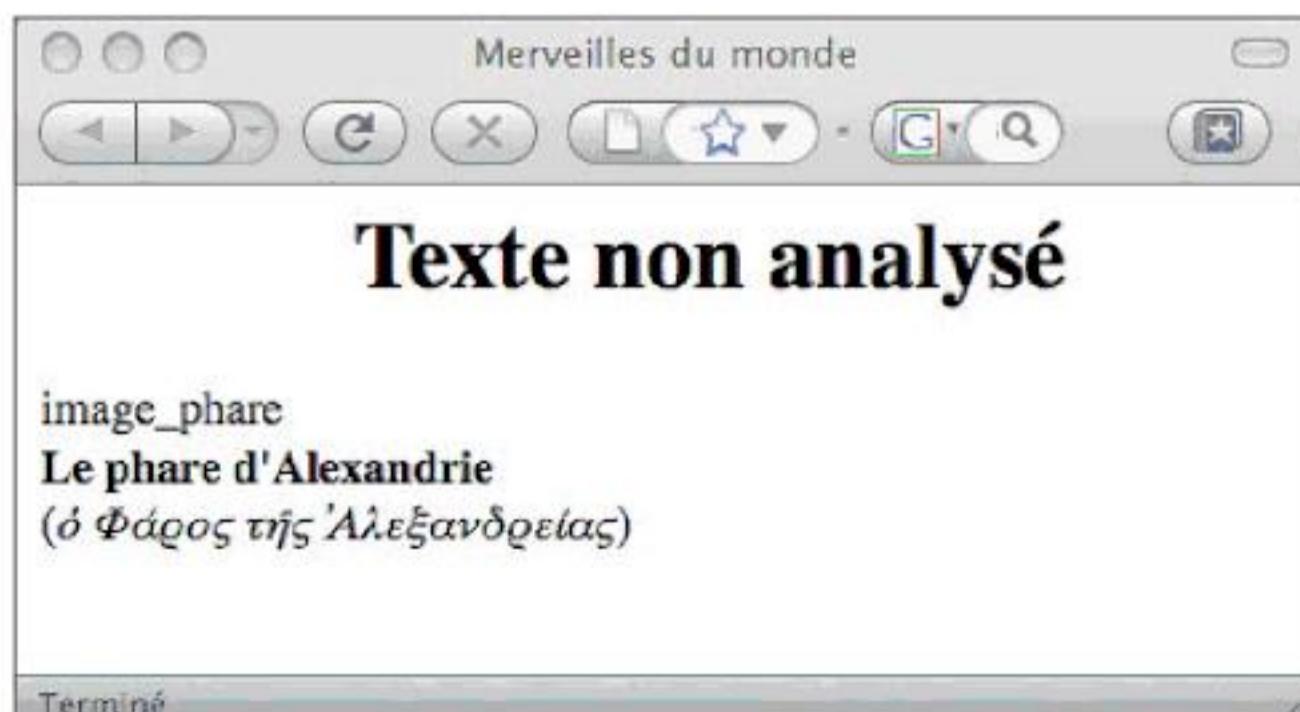
```

x 1 1

<?xml version="1.0" standalone="no"?>
...
<anciennes_merveilles>
    <merveille>
        <nom langue="Français">Le phare
        d'Alexandrie</nom>
        <nom langue="Grec">ὁ Φάρος τῆς Ἀλεξανδρείας
        </nom>
        <photo source="image_phare" />
    </merveille>
</anciennes_merveilles>

```

**Figure 7.12** La valeur de l'attribut `source` de l'élément `photo` correspond au nom de l'entité qui référence les données non analysées.



**Figure 7.13** Après la transformation XSLT du document XML de la Figure 7.12, le résultat affiché par notre navigateur est décevant, mais il le serait tout autant avec la plupart des autres navigateurs actuels.

## Intégrer un contenu non analysé

Vous pouvez maintenant intégrer dans votre document XML l'entité que nous venons de créer pour le contenu non analysé. Ces entités n'ont pas de références (contrairement aux entités analysées que nous avons vues précédemment) mais sont désignées par le type d'attribut `ENTITY` dans la déclaration `ATTLIST` de l'élément concerné (voir Figure 7.11). Vous pouvez également utiliser le type `ENTITIES` si vous souhaitez que l'attribut puisse contenir plusieurs références à des entités non analysées.

Vous pouvez ensuite associer un état ou une valeur par défaut à l'attribut, comme expliqué dans les sections "Définir des attributs" et "Définir des valeurs par défaut" du chapitre précédent.

Pour intégrer cette entité dans un document XML, il faut ajouter la directive `standalone="no"` dans la déclaration XML afin d'indiquer au processeur que ce document a besoin d'un fichier externe (celui qui contient l'entité non analysée). Pour intégrer l'entité dans le code XML, il suffit ensuite d'ajouter `nom_att="nom_ent"`, où `nom_att` est le nom de l'attribut de type `ENTITY` et `nom_ent` est le nom de l'entité NDATA (voir Figure 7.12).

### CONSEILS

- Bien que les processeurs XML soient supposés utiliser les informations de la notation pour les aider à afficher/exécuter l'entité non analysée, les navigateurs actuels en sont incapables. En d'autres termes, ils n'afficheront pas les données intégrées, comme vous pouvez le constater à la Figure 7.13.
- En réalité, tout le monde s'accorde à dire que les entités non analysées sont compliquées et confuses. Vous pourriez, par exemple, les remplacer en initialisant la valeur d'un élément avec une URL pointant vers le fichier de votre choix et utiliser ses attributs pour ajouter des informations supplémentaires.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

```
d t d
<!ELEMENT image_principale EMPTY>
<!ATTLIST image_principale
    fichier CDATA #REQUIRED
    l CDATA #REQUIRED
    h CDATA #REQUIRED>
```

**Figure 7.16** Extrait d'une DTD sauvegardée dans un fichier séparé nommé `image.ent`. Ce fichier sera utilisé par d'autres DTD et contient les déclarations de l'élément `image_principale` et de ses attributs.

```
d t d
<!ENTITY % image_complete SYSTEM "image.ent">
```

**Figure 7.17** On définit d'abord l'entité paramètre externe dans cette DTD et dans toutes celles qui déclarent l'élément `image_principale` et ses attributs.

```
d t d
<!ENTITY % image_complete SYSTEM "image.ent">

<!ELEMENT anciennes_merveilles (merveille+)>
<!ELEMENT merveille (nom+, lieu, hauteur,
historique, image_principale, source*)>
...
<!ELEMENT histoire (#PCDATA | para)*>
<!ELEMENT para EMPTY>
%image_complete;
<!ELEMENT source EMPTY>
<!ATTLIST source
    idsection CDATA #REQUIRED
    idjournal CDATA #REQUIRED>
```

**Figure 7.18** On peut ensuite utiliser l'entité en tapant sa référence, `%image_complete;`.

```
x m l
...
<image_principale fichier="gardens.jpg"
    l="528" h="349"/>
<source idsection="11" idjournal="24"/>
<source idsection="18" idjournal="151"/>
</merveille>
...
```

**Figure 7.19** Lors de l'analyse de la DTD de la Figure 7.18, le processeur remplace la référence d'entité `%image_complete;` par le contenu du fichier de la Figure 7.16. L'élément `image_principale` du fichier XML est désormais considéré comme valide par rapport à cette DTD.

## Créer une entité paramètre externe

Comme les entités générales, les entités paramètres peuvent être créées dans des fichiers externes.

Pour ce faire, comme pour les entités générales externes, on place le contenu de l'entité dans un fichier texte portant l'extension `.ent` (voir Figure 7.16) et on définit l'entité paramètre avec la syntaxe `<!ENTITY % nom_ent SYSTEM contenu.uri>`.

### CONSEILS

- L'utilisation d'une entité paramètre externe est rigoureusement identique à celle d'une entité paramètre interne.
- Si l'on utilise une DTD interne (voir le chapitre suivant), il faut ajouter `standalone="no"` dans la déclaration XML de cette DTD afin d'indiquer au processeur que le document XML a besoin d'un fichier externe.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

# Index

## Symboles

---

#FIXED, syntaxe DTD [86](#)  
&#, entité 14  
&#, entité XML prédéfinie [92](#)  
&apos;, entité 14  
&apos;, entité XML prédéfinie [92](#)  
&gt;, entité 14  
&gt;, entité XML prédéfinie [92](#)  
&lt;, entité 14  
&lt;, entité XML prédéfinie [92](#)  
&quot;, entité 14  
&quot;, entité XML prédéfinie [92](#)  
(#PCDATA), syntaxe DTD 77  
<!ELEMENT, syntaxe DTD 77  
<xsl:  
    value-of>, élément XSLT 26

## A

---

Addition 52, 58  
Ajax 220, 238  
all, élément de XML Schema 142, 149  
analyze-string, fonction XPath 203  
and, opérateur booléen 50  
annotation, élément de XML Schema 116  
Annotation de XML Schema 116  
ANSI, encodage 250  
ANY, syntaxe DTD [83](#)  
anyType, type de XML Schema 136, 137, 140, 145, 146

anyURI, type de XML Schema 118  
apply-template, élément de XSLT 183  
ASCII, encodage 250  
ATTLIST, syntaxe DTD 85  
Attribut 4, 6, [11](#), [84](#)  
    axe 44  
    d'un nœud 44  
    définir 85, 152  
    exiger 153  
    groupes  
        définir 155  
        référencer 156  
    prédefinir le contenu 154  
    production 33  
    valeur 5  
attribute, élément de XML Schema 144, 145, 146, 147, 152  
attributeGroup, élément de XML Schema 155, 156  
avg(), fonction XPath 194  
Axe  
    des attributs 44  
    XPath 44

## B

---

Bases de données 215  
Bien formé, document 5  
Bitmap *voir* Images  
boolean, type de XML Schema 118  
border-size, attribut XSL-FO 65  
break-after, attribut XSL-FO 69  
break-before, attribut XSL-FO 69, 70

## C

---

Caractères Unicode 253  
Casse, modifier 57  
CDATA, section [15](#)  
ceiling(), fonction de XSLT 187  
channel, élément RSS 225  
Chemins d'accès 38  
  création 46  
choice, élément de XML Schema 143  
collection(), fonction XQuery 215  
column-count, attribut XSL-FO 70  
column-gap, attribut XSL-FO 70  
Commentaires 13, 202  
complexContent, élément de XML Schema 137  
complexType, élément de XML Schema 138, 139, 140  
Condition  
  quantifier 198  
  tester 197  
contains(), fonction XPath 58  
content-height, attribut XSL-FO 65  
current-date, fonction XPath 201  
current-group(), fonction de XSLT 188

## D

---

date, type de XML Schema 118, 120  
dateTime, type de XML Schema 120  
decimal, type de XML Schema 118, 122  
Déclaration  
  de type 103  
  XML 4, [7](#)  
default, attribut de XML Schema 123, 154  
definitions, élément WSDL 230  
distinct-values, fonction XPath 199  
Division 52  
doc(), fonction XQuery 208  
DOCTYPE, déclaration de type 103, 104  
Document  
  bien formé 5  
  création [3](#)

## DTD 76

entité [92](#)  
espace de noms 177  
externe  
  créer 102  
    déclarer 103  
  externe publique  
    déclarer 107  
  nommer 106  
  interne 104  
duration, type de XML Schema 120

## E

---

Élément 6  
  de type simple 118  
  fils 4, 9  
    ordre quelconque 142  
    séquence 141  
  imbrication 10  
  racine 4, 5, 8  
  vide 5, 12, 145  
element, élément de XML Schema 118  
elementFormDefault, attribut de XML Schema 172, 173  
EMPTY, syntaxe DTD 78  
enclosure, élément RSS 225  
Encodage des caractères 250  
encoding, attribut XML 250  
Entités 14  
  DTD [92](#)  
  références 252  
  & 14  
  ' 14  
  > 14  
  < 14  
  " 14  
ENTITIES, syntaxe DTD [97](#)  
ENTITY, syntaxe DTD [92, 97, 99](#)  
enumeration, facette de XML Schema 128  
Envelope, élément SOAP 229  
ePub 236, 239

**Espace de noms** 114, 162

- DTD 177
- par défaut 163
- schéma 176
- préfixe de nom 164
- remplir 168
- XSL-FO [63](#)
- XSLT 178

**Espaces** 6

- every, instruction XPath** 198
- exclude-result-prefixes, attribut de XSLT** 186
- exists, fonction XPath** 198

**Expressions**

- conditionnelles 212
- FLWOR 210
- XPath 209

## F

---

**Facette, XML Schema** 126

**Feuille de style**

- simplifiée avec XSLT 2.0 183
- XSLT 22

**Fils, élément** 4, 9

**fixed, attribut de XML Schema** 123, 154

**float, type de XML Schema** 122

**fo:**

- block, élément XSL-FO [62, 64, 67](#)
- external-graphic, élément XSL-FO 65
- flow, élément XSL-FO [71](#)
- layout-master-set, élément XSL-FO [63, 66](#)
- page-number, élément XSL-FO 69
- page-sequence, élément XSL-FO [63, 67, 71](#)
- region-after, élément XSL-FO 69
- region-before, élément XSL-FO [67](#)
- region-body, élément XSL-FO [71](#)
- root, élément XSL-FO 62, 66
- simple-page-master, élément XSL-FO [71](#)

**Fonctions**

- appeler 187
- création 186
- utilisateur 214

**font-size, attribut XSL-FO** 64

**for, clause XQuery** 210

**for-each-group, élément de XSLT** 188

**for-in-return, instruction XPath** 200

**format-number(), fonction XPath** 54, [55](#)

**format-time, fonction XPath** 201

**Formatage chaîne de caractères** 196

**FPI (Formal Public Identifier)** 106, 107

**function, élément de XSLT** 186, 187

## G

---

**gDay, type de XML Schema** 121

**gMonth, type de XML Schema** 121

**gMonthDay, type de XML Schema** 121

**group, élément de XML Schema** 150, 151

**Groupe modèle**

- d'un type composé 140

- nommé

- définir 150

- référencer 151

**gYear, type de XML Schema** 121

**gYearMonth, type de XML Schema** 121

## H - I

---

**Header, élément SOAP** 229

**HTML, production** 24

**id(), fonction XPath** 58

**ID, syntaxe DTD** [88, 90](#)

**IDREF, syntaxe DTD** [89](#)

**IDREFS, syntaxe DTD** [89](#)

**if-then-else**

- instruction XPath 197

- instruction XQuery 212

**Image, ajouter** 65

**Imbrication des éléments** 10

**import, élément de XML Schema** 175

**import-schema, élément de XSLT** 189

**include, élément de XML Schema** 174

## Instruction de traitement [7](#)

xmlstylesheet 20  
xsl:output 24  
**int**, type de XML Schema 122  
**integer**, type de XML Schema 122  
**ISO** (*International Organization for Standardization*) 106  
**ISO-8859-1**, encodage 250

## L

---

**KML** (*Keyhole Markup Language*) 232, 239  
**kml**, élément de KML 233  
**last()**, fonction XPath [51](#)  
**length**, facette de XML Schema 129  
**list**, élément de XML Schema 133

## M

---

**manifest**, élément ePub) 237  
**margin**, attribut XSL-FO 66  
**match**, fonction XPath 203  
**max()**, fonction XPath 195  
**maxExclusive**, facette de XML Schema 127  
**maxInclusive**, facette de XML Schema 126  
**maxOccurs**, attribut de XML Schema 141, 142, 143, 146  
**Métadonnées** 6  
**method**, attribut de XSLT 185  
**min()**, fonction XPath 195  
**minExclusive**, facette de XML Schema 127  
**minLength**, facette de XML Schema 129  
**minOccurs**, attribut de XML Schema 141, 142, 143, 149  
**Minuteurs** *voir* Timer  
**Modèle de page** 66  
    ajouter [71](#)  
    en-tête [67](#)  
**Module RSS** 226  
**Multiplication** 52

## N

---

**name()**, fonction XPath 58  
**NDATA**, syntaxe DTD [96](#)  
**NMTOKEN**, syntaxe DTD [90](#)  
**NMTOKENS**, syntaxe DTD [90](#)  
**Nombres**  
    arrondir [55](#)  
    formater 54  
    opérations 52  
**noNamespaceSchemaLocation**, attribut XML Schema 115  
**normalize-space()**, fonction XPath [59](#)  
**not()**, fonction XPath [59](#)  
**not**, fonction XPath 50  
**NOTATION**, syntaxe DTD [96](#)

## Nœud

    attribut 44  
    boucles 28  
    compter 53  
    courant 40  
    descendants [47](#)  
    du document 38  
    fils d'un 42  
    localisation 38  
    traitement conditionnel 30

## O

---

**OCF** (*OEBPS Container Format*) 237  
**ODF** 234, 239  
**OGC** (*Open Geospatial Consortium*) 232  
**OOXML** 235, 239  
**Opacité** *voir* Transparence  
**OpenXML Writer**, format OOXML 235  
**OPF** (*Open Package Format*) 237  
**OPS** (*Open Publication Structure*) 237  
**or**, opérateur booléen 50  
**order**, clause XQuery 211

**P****Page**

- colonne 70
  - en-tête [67](#)
  - saut de page 69
- page-height**, attribut XSL-FO 66
- page-width**, attribut XSL-FO 62, 66
- pattern**, facette de XML Schema 130
- Placemark**, élément de KML 233
- Polygon**, élément de KML 233
- positiveInteger**, type de XML Schema 122
- Prédicat XPath** 45

**Q - R****Quantificateurs, syntaxe DTD** 81**Racine**

- élément 4, 5, 8
  - règle XSLT 21
- ref**, attribut de XML Schema 148, 152

**Références**

- d'entités 252
- numériques de caractères 251

**Règle racine, XSLT** 21

- création [23](#)
- par défaut [23](#)

**replace()**, fonction XPath 57**replace**, fonction XPath 203**restriction**, élément de XML Schema 124**result-document**, élément de XSLT 184, 185**rootfile**, élément ePub 236**round()**, fonction XPath [55](#)**RSS** 224, 238

- étendre 226
- module 226

**rss**, élément RSS 225**S**

- schema**, élément de XML Schema 148, 150, 155
- schema**, élément racine de XML Schema 114
- schemaLocation**, attribut de XML Schema 175
- sequence**, élément de XML Schema 141
- Séquence**, parcourir 200
- simpleContent**, élément de XML Schema 137, 144
- simpleType**, élément de XML Schema 124, 125, 152
- SOAP** 228, 238
- some**, instruction XPath 198
- sort**, élément de XSLT 188
- Sous-chaîne**, extraction 56
- Soustraction** 52
- space-after**, attribut XSL-FO 64
- span**, attribut XSL-FO 70
- standalone**, directive XML [95, 97, 99](#)
- string**, type de XML Schema 118
- string-length()**, fonction XPath 58
- stylesheet**, élément de XSLT 183
- styleURL**, élément de KML 233
- substring()**, fonction XPath 56
- substring-after()**, fonction XPath 56
- substring-after**, fonction XPath 199
- sum()**, fonction XPath 58, 195

**Syntaxe (éléments)**

- XML 259
- XPath 261
- XSD 260
- XSL 261

**SYSTEM**, syntaxe DTD [94, 96, 99](#)**T**

- targetNamespace**, attribut de XML Schema 168
- targetNamespace**, attribut XML 170
- template**, élément de XSLT 183
- time**, type de XML Schema 118, 120

**tokenize, fonction XPath** 203  
**totalDigits, facette de XML Schema** 132  
**translate(), fonction XPath** 196  
**transtypage, opérateur XSLT** 201  
**type, attribut de XML Schema** 139  
**Types**  
    composés 136  
        anonymes 138  
        avec éléments fils 140  
        nommés 139  
    liste 133  
    nommer 125  
    numériques 122  
    simples  
        créer 124  
        définir 118  
    temporels 120  
    union 134  
**types, élément WSDL** 230

## U

---

**UBR (*UDDI Business Registry*)** 231  
**UDDI** 231  
**unbounded, voir maxOccurs** 149  
**Unicode** 253  
**union, élément de XML Schema** 134  
**unparsed-text, fonction XPath** 203  
**upper-case(), fonction XPath** 196  
**URI (*Uniform Resource Identifier*)** 162  
**URL (*Uniform Resource Locator*)** 162  
**use, attribut de XML Schema** 153, 154  
**UTF-8, encodage** 250

## V

---

**Valeur**  
    comparaison 50  
    maximale 195  
    minimale 195  
    moyenne 194  
    textuelle d'un noeud XML 26  
**Validateur en ligne** 105  
**value, attribut de XML Schema** 153  
**version, attribut de XSLT** 182  
**Vide, élément** 5, 12

## W

---

**WordProcessingML** 235  
**WSDL** 230, 238

## X

---

**XHTML** 184  
**XML**  
    création de documents 3  
    déclaration 4, 7  
    éditeurs 244  
    éléments de syntaxe 263  
    outils 247  
    spécifications 247  
**xmlstylesheet, instruction de traitement** 20  
**XMLHttpRequest, objet** 220, 221  
**xmlns, attribut XML** 169, 170

**XML Schema** 112

- annoter 116
- associer à un document XML 115
- espace de noms 175
- répartition dans plusieurs fichiers 174

**XPath** 37

- axe 44
- éléments de syntaxe 261
- expressions dans XQuery 209
- fonctions 49
- prédictat 45

**XPath 2.0** 191**XQuery 1.0** 205**XSD, éléments de syntaxe** [264](#)**XSL, éléments de syntaxe** 261**XSL-FO** 61

- créer un document [63](#)
- espace de noms [63](#)

**xsl:**

- apply-templates, élément XSLT 34, 35, 40, 68
- attribute, élément XSLT 33
- choose
  - élément XSLT 30
  - instruction XSLT 31
- for-each, élément XSLT 28, 32, 40
- if, élément XSLT 30, 51
- otherwise, élément XSLT 31
- output, élément XSLT 68
- output, instruction de traitement 24
- sort, élément XSLT 32
- template, élément XSLT 34, 40, 68
- text, élément XSLT 42
- validation, attribut de XSLT 189
- when, élément XSLT 31

**XSLT** 19

- espace de noms 178
- feuille de style 22

**XSLT 2.0** 182



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.