



XML精要

快速参考手册

—— XML, XPath, XSLT, XML Schema, SOAP

“如果您要开发与XML相关的应用程序，请将本书放进您的工具箱里，它非常有用。”

—— 微软公司.NET XML Framework项目经理Mark Fussell

[美] Aaron Skonnard 著
Martin Gudgin

牛韬 英宇 译

769

2005.11.12
C21

XML 精要快速参考手册

——XML、XPath、XSLT、XML Schema、SOAP

[美] Aaron Skonnard, Martin Gudgin 著
牛 韬 英 宇 译



A1013251

人 民 邮 电 出 版 社

图书在版编目(CIP)数据

XML 精要快速参考手册: XML、XPath、XSLT、XML Schema、SOAP / (美)斯科那德(Skonnard, A.), (美)古德金(Gudgin, M.) 著; 牛韬, 英字译. —北京: 人民邮电出版社, 2002.10
ISBN 7-115-10571-5

I. X... II. ①斯...②古...③牛...④英... III. 可扩充语言, XML—程序设计—手册
IV. TP312-62

中国版本图书馆 CIP 数据核字(2002)第 062665 号

版权声明

Simplified Chinese Copyright © 2002 by PEARSON EDUCATION NORTH ASIA LIMITED and POSTS & TELECOMMUNICATIONS PRESS.

Essential XML Quick Reference: a programmer's reference to XML, XPath, XSLT, XML Schema, SOAP, and more
By Aaron Skonnard, Martin Gudgin

Copyright © 2002

All Rights Reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Addison-Wesley

This edition is authorized for sale only in People's Republic of China (excluding the Special Administrative Region of Hong Kong and Macao).

本书封面贴有 Pearson Education 出版集团激光防伪标签, 无标签者不得销售。

XML 精要快速参考手册

——XML、XPath、XSLT、XML Schema、SOAP

◆ 著 [美] Aaron Skonnard Martin Gudgin
译 牛 韬 英 字
责任编辑 陈 昇

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
读者热线 010-67132705
北京汉魂图文设计有限公司制作
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ 开本: 787×1092 1/16
印张: 18
字数: 426 千字 2002 年 10 月第 1 版
印数: 1-4 000 册 2002 年 10 月北京第 1 次印刷

著作权合同登记 图字: 01-2002-3736 号

ISBN 7-115-10571-5/TP·3045

定价: 33.00 元

本书如有印装质量问题, 请与本社联系 电话: (010) 67129223

前 言

对于任何人,本书都和当前主流 XML 技术一起运用。本书设计成回答许多公共的 XML 相关技术问题的方便而完整的快速参考手册。

本书完全覆盖每一个主题,拥有大量有用示例的传统软件包参考设计。每一章都有概要介绍,并伴随着详细的参考信息。

本书采用的方法是假设读者已经基本理解书中给定的每一个主题。

目 录

第 1 章 XML 1.0 与命名空间	1
1.1 元素	1
1.2 元素、命名空间和命名空间声明	2
1.3 属性	4
1.4 属性和命名空间	4
1.5 处理指令	5
1.6 注释	5
1.7 空白	6
1.8 禁用的字符常量	7
1.9 CDATA 段	8
1.10 XML 声明	8
1.11 字符引用	9
1.12 良构的 XML	10
1.13 参考	10
第 2 章 文档类型定义	11
2.1 DTD 简介	11
2.2 DOCTYPE	11
2.2.1 内部声明	12
2.2.2 外部声明	12
2.2.3 内部和外部声明	13
2.3 ELEMENT	14
2.4 ATTLIST	16
2.5 ENTITY	18
2.5.1 内部参数实体	19
2.5.2 外部参数实体	21
2.5.3 内部通用实体	22
2.5.4 外部通用解析实体	22
2.5.5 非解析实体	23
2.6 NOTATION	23
2.7 INCLUDE 和 IGNORE	24

2.8 参考	25
第3章 XPath 1.0	26
3.1 XPath 简介	26
3.2 定位路径表达式	29
3.2.1 定位步骤	30
3.2.2 轴	30
3.2.3 节点测试	33
3.2.4 谓词	34
3.2.5 定位路径缩写	35
3.3 基本表达式	35
3.3.1 布尔表达式	36
3.3.2 等式表达式	36
3.3.3 关系表达式	37
3.3.4 数值表达式	38
3.4 核心函数库	38
3.4.1 boolean	39
3.4.2 ceiling	40
3.4.3 concat	40
3.4.4 contains	41
3.4.5 count	41
3.4.6 false	41
3.4.7 floor	42
3.4.8 id	42
3.4.9 lang	42
3.4.10 last	43
3.4.11 local-name	43
3.4.12 name	43
3.4.13 namespace-uri	44
3.4.14 normalize-space	44
3.4.15 not	44
3.4.16 number	45
3.4.17 position	45
3.4.18 round	46
3.4.19 starts-with	46
3.4.20 string	46
3.4.21 string-length	47
3.4.22 substring	47
3.4.23 substring-after	48

3.4.24	substring-before	48
3.4.25	sum	48
3.4.26	translate	49
3.4.27	true	49
3.5	参考	49
第 4 章	XPointer、XInclude 和 XML Base	50
4.1	XPointer 1.0	50
4.1.1	完整形式	50
4.1.2	无修饰名称	51
4.1.3	子节点序列	52
4.1.4	XPointer 扩充到 XPath	52
4.1.5	XPointer 节点测试	53
4.1.6	XPointer 函数库	54
4.2	XInclude	57
4.3	XML Base	59
4.4	参考	60
第 5 章	XSL Transformations 1.0	61
5.1	XSLT 编程简介	61
5.2	XSLT 类型和表达式	64
5.3	模式	65
5.4	冲突解决	66
5.5	内建模板	67
5.6	基于原型的转换语法	68
5.7	属性值模板	68
5.8	空白	69
5.9	元素库	69
5.9.1	apply-imports	72
5.9.2	apply-templates	73
5.9.3	attribute	74
5.9.4	attribute-set	76
5.9.5	call-template	77
5.9.6	choose	78
5.9.7	comment	79
5.9.8	copy	79
5.9.9	copy-of	80
5.9.10	decimal-format	81
5.9.11	element	82

5.9.12	fallback	83
5.9.13	for-each	84
5.9.14	if	85
5.9.15	import	85
5.9.16	include	86
5.9.17	key	87
5.9.18	message	89
5.9.19	namespace-alias	90
5.9.20	number	91
5.9.21	otherwise	93
5.9.22	output	93
5.9.23	param	95
5.9.24	preserve-space	96
5.9.25	processing-instruction	97
5.9.26	sort	97
5.9.27	strip-space	98
5.9.28	stylesheet	99
5.9.29	template	99
5.9.30	text	102
5.9.31	transform (stylesheet)	103
5.9.32	value-of	104
5.9.33	variable	105
5.9.34	when	107
5.9.35	with-param	108
5.10	XSLT 函数库	108
5.10.1	current	109
5.10.2	document	109
5.10.3	element-available	111
5.10.4	format-number	112
5.10.5	function-available	112
5.10.6	generate-id	113
5.10.7	key	113
5.10.8	system-property	113
5.10.9	unparsed-entity-uri	114
5.11	参考	114
第 6 章	SAX 2.0	116
6.1	SAX UML 快速参考	116
6.2	SAX 的接口和类	116

6.2.1	Attribute	119
6.2.2	ContentHandler	121
6.2.3	DTDHandler	127
6.2.4	EntityResolver	129
6.2.5	ErrorHandler	130
6.2.6	Locator	132
6.2.7	XMLFilter	134
6.2.8	XMLReader	135
6.3	功能和特性	139
6.4	参考	140
第 7 章	DOM Level 2	141
7.1	DOM UML	141
7.2	DOM 接口	141
7.2.1	Attr	141
7.2.2	CDATASection	143
7.2.3	CharacterData	144
7.2.4	Comment	146
7.2.5	Document	146
7.2.6	DocumentFragment	149
7.2.7	DocumentType	149
7.2.8	DOMImplementation	150
7.2.9	Element	151
7.2.10	Entity	154
7.2.11	EntityReference	154
7.2.12	NamedNodeMap	154
7.2.13	Node	156
7.2.14	NodeList	161
7.2.15	Notation	163
7.2.16	ProcessingInstruction	163
7.2.17	Text	164
7.3	参考	164
第 8 章	XML Schema 数据类型	165
8.1	数据类型分组	165
8.2	数据类型	167
8.2.1	anyURI	167
8.2.2	base64Binary	168
8.2.3	boolean	169

8.2.4	byte	169
8.2.5	date	170
8.2.6	dateTime	170
8.2.7	decimal	171
8.2.8	double	172
8.2.9	duration	172
8.2.10	ENTITIES	173
8.2.11	ENTITY	174
8.2.12	float	174
8.2.13	gDay	175
8.2.14	gMonth	175
8.2.15	gMonthDay	176
8.2.16	gYear	176
8.2.17	gYearMonth	177
8.2.18	hexBinary	177
8.2.19	ID	178
8.2.20	IDREF	178
8.2.21	IDREFS	178
8.2.22	int	179
8.2.23	integer	179
8.2.24	language	180
8.2.25	long	181
8.2.26	Name	181
8.2.27	NCName	182
8.2.28	negativeInteger	182
8.2.29	NMTOKEN	183
8.2.30	NMTOKENS	183
8.2.31	nonNegativeInteger	184
8.2.32	nonPositiveInteger	184
8.2.33	normalizedString	185
8.2.34	NOTATION	185
8.2.35	positiveInteger	186
8.2.36	QName	187
8.2.37	short	187
8.2.38	string	188
8.2.39	time	189
8.2.40	token	189
8.2.41	unsignedByte	190
8.2.42	unsignedInt	190

8.2.43	unsignedLong	191
8.2.44	unsignedShort	192
8.3	Facet	192
8.3.1	enumeration	193
8.3.2	fractionDigits	193
8.3.3	length	194
8.3.4	maxExclusive	195
8.3.5	maxInclusive	195
8.3.6	maxLength	196
8.3.7	minExclusive	197
8.3.8	minInclusive	198
8.3.9	minLength	198
8.3.10	pattern	199
8.3.11	totalDigits	200
8.3.12	whiteSpace	201
8.4	语言结构	201
8.4.1	simpleType	202
8.4.2	restriction	202
8.4.3	list	203
8.4.4	union	204
8.5	参考	205
第 9 章	XML Schema 结构	207
9.1	Schema 元素分组	207
9.2	结构	210
9.2.1	all	210
9.2.2	annotation	211
9.2.3	any	212
9.2.4	anyAttribute	214
9.2.5	appinfo	216
9.2.6	attribute	217
9.2.7	attributeGroup	219
9.2.8	choice	221
9.2.9	complexContent	222
9.2.10	complexType	222
9.2.11	documentation	225
9.2.12	element	225
9.2.13	extension	229
9.2.14	field	231

9.2.15	group	232
9.2.16	import	234
9.2.17	include	235
9.2.18	key	236
9.2.19	keyref	238
9.2.20	notation	238
9.2.21	redefine	239
9.2.22	restriction	241
9.2.23	schema	243
9.2.24	selector	246
9.2.25	sequence	247
9.2.26	simpleContent	248
9.2.27	unique	249
9.3	XML Schema 结构: 实例属性	250
9.3.1	nil	250
9.3.2	noNamespaceSchemaLocation	251
9.3.3	SchemaLocation	251
9.3.4	type	252
9.4	参考	253
第 10 章	SOAP 1.1	254
10.1	SOAP 消息简介	254
10.2	SOAP 消息中的元素	255
10.2.1	Body	255
10.2.2	Envelope	256
10.2.3	Fault	256
10.2.4	Header	258
10.3	SOAP 消息中的属性	259
10.3.1	actor	259
10.3.2	encodingStyle	260
10.3.3	mustUnderstand	260
10.4	SOAP 串行化规则简介	261
10.4.1	简单结构化数据的串行化	262
10.4.2	带有多引用结构化数据的串行化	264
10.4.3	在复杂数据结构中处理空引用	265
10.4.4	串行化动态分类数据	265
10.4.5	数组	267
10.4.6	多维数组	267
10.4.7	数组的部分传输	268

10.4.8 稀疏数组	269
10.4.9 复合数组	269
10.5 SOAP 和 HTTP 绑定介绍	271
10.5.1 Content-Type	271
10.5.2 Content-Length	271
10.5.3 SOAPAction	272
10.6 参考	272

第 1 章 XML 1.0 与命名空间

XML 1.0 和 XML 中的命名空间为结构化数据和将标记应用于文档提供了基于标记的语法。符合 XML 1.0 和 XML 规范的命名空间的文档，可能由多种语法结构组成，比如元素、命名空间声明、属性、处理指令、注释和文本。本章提供了 XML 中的每种结构化元素及其各自语法。

1.1 元 素

```
<tagname></tagname>
<tagname/>
<tagname>children</tagname>
```

通常，元素典型地组成了 XML 文档中的大部分内容。每个 XML 文档都准确地有一个顶层的元素，也就是我们所知的**文档元素** (*document element*)。元素有名字，也可能有后代。这些后代可能本身就是元素或者是处理指令、注释、字符数据 (CDATA) 段或者字符。元素的后代是有序的。元素也用属性来描述。元素的属性是无序的。元素可能有与之关联的命名空间声明。元素的命名空间声明是无序的。

元素由一对标记串行化而成：起始标记和终止标记。起始标记的语法是一个小于号 (<)，后面紧跟元素名称，也就是紧跟着我们所说的**标记名** (*tagname*)，然后跟着一个大于号 (>)。终止标记的语法是字符序列</，后面紧跟着标记名，然后跟着大于号 (>)。元素的后代则串行化在父元素的起始标记和终止标记之间。在元素没有后代的情况下，元素称为**空元素** (*empty*)。有一种速记语法可以用于表示空元素，就是由一个小于号紧跟着标记名和字符序列/>组成。

XML 并没有定义任何元素名称，它允许 XML 的设计者来选择要使用的名称。XML 中的元素名称是区分大小写的，它必须开始于字母或者下划线 (_)。然后可以紧跟任意长度的字母、数字、句点 (.)、连字符 (-)、下划线 (_) 和冒号。但是，由于冒号在 XML 中还用于命名空间的语法中，因此它们不应该用于元素名称中，除非在其规范中做了描述 (参看第 1.2 节)。以字符序列 xml 开始的元素名称，或者关于它的任何调整，在 XML 的规范里都被保留以待将来使用。

例子

带有后代的元素

```
<Person>
  <name>Martin</name>
  <age>33</age>
</Person>
```

元素的标记名是 **Person**。这个元素有两个后代，标记名分别叫作 **name** 和 **age**。这两个后代元素都带有文本内容。

空元素

```
<Paid></Paid>
```

标记名为 **Paid** 的空元素。

速记的空元素

```
<Paid/>
```

使用速记语法的标记名叫作 **Paid** 的空元素。

1.2 元素、命名空间和命名空间声明

```
<prefix:localname xmlns:prefix='namespace URI' />
<prefix:localname xmlns:prefix='namespace URI'></
  prefix:localname/>
<prefix:localname xmlns:prefix='namespace URI'>children</
  prefix:localname/>
```

因为 XML 允许设计者自己选择标记名，所以就可能有两个或者更多的设计者为某些元素或者所有元素选择相同的标记名。XML 命名空间提供了一种方法来明确地区分这些 XML 元素：有相同的本地名称，但实际上来自不同的词汇表。这就要将元素与命名空间进行关联来实现。命名空间担当着与之关联的所有元素的作用域。命名空间本身也有名称。命名空间的名称是统一资源标识符（Uniform Resource Identifier, URI）。这种 URI 是一个惟一的字符串，不能够废除。命名空间的名称和元素的本地名称合在一起有一个全局惟一的名字，就是我们知道的**限定名（qualified name）**。

命名空间声明一般出现在元素的起始标记中，通常习惯于把一个命名空间的名称映射为另一个通常很短的字符串，也就是我们所知的**命名空间前缀（namespace prefix）**。命名空间声明的语法为：`xmlns:prefix='URI'`。无前缀使用默认命名空间声明映射命名空间也是可能的。默认命名空间声明的语法是：`xmlns='URI'`。在这两种情况下，URI 都出现在单引号（'）或者双引号（"）中。一个元素只能有一个默认命名空间声明出现。但是非默认的命名空间声明则是无限数目的，并且可以为相同的 URI 提供不同的前缀部分。尽管把相同 URI 映射到多个前缀并不是特别有用，但它是合法的。

所有的命名空间声明都有作用域，这个作用域就是它适用的一套元素。命名空间声明的作用域是它声明的元素以及元素的后代。在作用域内给后代元素所使用的前缀提供一个新的映射可以取代给定前缀到命名空间名称的映射。给后代元素提供默认的命名空间声明可以取代在作用域内的默认命名空间。

文档里的所有符合 XML 规范中命名空间的元素名称叫作 QName。从语法上来说，所有 QName 都有本地名和可选前缀。本地名和前缀合称为 NCName。NCName 是一个不带冒号的名字。有前缀的元素的语法是：前缀，后面跟着冒号，然后跟着本地名。赋予前缀的元素的命名空间就是作用域内为该前缀声明所指定的命名空间。如果在作用域里没有这样的命名空间声明就是错误的。如果无前缀元素的命名空间存在的话，该命名空间是作用域

内默认的命名空间声明所指定的命名空间。如果作用域中没有默认命名空间声明，则该元素不存在于任何命名空间中。不存在于任何命名空间中的元素被称作**非限定元素** (*unqualified elements*)。非限定元素的命名空间名称是空字符串“”。如果默认命名空间声明在作用域中，并且需要有非限定元素，则可以为元素提供 `xmlns= ''` 这样的形式来屏蔽默认命名空间声明。

例子

限定的元素名和非限定的元素名

```
<pre:Person xmlns:pre='urn:example-org:People' >
  <name>Martin</name>
  <age>33</age>
</pre:Person>
```

例子中元素的本地名叫作 `Person`，前缀 `pre` 是从命名空间 `urn: example-org: People` 映射来的。元素 `Person` 有两个后代，本地名为 `name` 和 `age`，它们都是非限定元素，因为它们不存在于任何的命名空间中。

使用默认命名空间声明的限定和非限定元素

```
<Person xmlns='urn:example-org:People' >
  <name xmlns=''>Martin</name>
  <age xmlns=''>33</age>
</Person>
```

元素的本地名叫作 `Person`，没有前缀。根据作用域内该 URI 的默认命名空间声明元素存在于命名空间 `urn: example-org: People` 里。元素有两个后代，本地名分别叫作 `name` 和 `age`。这两个子元素都是非限定的，也就是说，他们不存在于任何的命名空间里。这个例子和前面的例子是等同的。

限定元素

```
<pre:Person xmlns:pre='urn:example-org:People' >
  <pre:name>Martin</pre:name>
  <pre:age>33</pre:age>
</pre:Person>
```

该元素的本地名是 `Person`，前缀是 `pre`，是从命名空间 URI `urn: example-org: People` 映射来的。元素有两个本地名为 `name` 和 `age` 的后代。这两个子元素都有同一个前缀 `pre`，都存在于命名空间 `urn: example-org: People`。

使用默认命名空间声明的限定元素

```
<Person xmlns='urn:example-org:People' >
  <name>Martin</name>
  <age>33</age>
</Person>
```

元素的本地名叫作 `Person`，没有前缀。根据作用域内该 URI 的默认命名空间声明元素存在于命名空间 `urn: example-org: People` 里。元素有两个本地名为 `name` 和 `age` 的后代。这两个子元素都存在于命名空间 `urn: example-org: People` 之中。这个例子和前面的例子是等同的。

1.3 属 性

```
name='value'
name="value"
```

元素可以用属性来注释。属性通常用来编码实际数据或者为元素提供元数据，也就是说，提供关于元素所显示内容的额外信息。给定元素的属性被串行化在元素的起始标记中。属性表示为用等号(=)分开的名/值对。属性名与元素名有相同的构造规则。属性值是自然的文本形式，必须出现在单引号或者双引号的内部。元素可以有任意数目的属性，但是它们的名称必须不同。

例子

数据属性

```
<Person name='Martin' age='33' />
```

本例中的 Person 使用属性来表示要比使用子元素更加清楚。

元数据属性

```
<age base='16' units='years' >20</age>
<age base="10" units="years" >32</age>
```

带有元数据属性的元素。

1.4 属性和命名空间

```
prefix:localname='value'
prefix:localname="value"
```

属性名称都是 QName。赋予前缀的属性的命名空间就是作用域内为该前缀声明所指定的命名空间。如果在作用域里没有这种命名空间声明是错误的。即使在作用域中有默认命名空间声明，无前缀属性也是不存在于任何命名空间中的。

例子

限定属性

```
<Person xmlns='urn:example-org:People'
  xmlns:b='urn:example-org:People:base'
  xmlns:u='urn:example-org:units' >
  <name>Martin</name>
  <age b:base='10' u:units='years' >33</age>
</Person>
```

本地名叫作 base 的属性存在于命名空间 urn: example-org: People: base 中；本地名叫作 units 的属性存在于命名空间 urn: example-org: units 中。