

国外计算机科学经典教材

DEFINITIVE XML Schema

XML

模式权威教程

Priscilla Walmsley 著
陈维军 乔安平 英宇 译



- 透视XML 模式的强大功能
- 完全涵盖了已获广泛认可的、W3C推荐的XML 标准
- 全面、实际地设计XML 模式
- 帮助富有经验的 DTD 开发人员迁移到XML 模式
- 由 W3C XML 模式工作组的成员——Priscilla Walmsley 编写，具有极高的权威性



清华大学出版社

<http://www.tup.tsinghua.edu.cn>



XML 模式权威教程

Priscilla Walmsley 著

陈维军 乔安平 英宇 译

清华大学出版社

北京市版权局著作权合同登记号：01-2002-3006

内 容 简 介

本书介绍的是一种功能强大的文档模式语言：XML Schema，它的独特功能包括强类型、模块化、继承以及一致性约束等。本书主要内容包括：XML Schema 如何为 XML 文档结构、内容和数据类型建模提供严格而完整的标准，XML Schema 的元素、属性和类型等构件，以及类型派生、模型组、替换组等高级内容。

本书适合于 XML 开发人员、系统架构师等专业人员及相关的计算机知识爱好者阅读。

Definitive XML Schema

Priscilla Walmsley

Copyright © 2002 by PH PTR.

Original English Language Edition Published by PH PTR.

All Rights Reserved.

本书中文简体字版由 Pearson Education 授权清华大学出版社出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，翻印必究。

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。

图书在版编目(CIP)数据

XML 模式权威教程/(美)沃姆斯利著；陈维军，乔安平，英宇译.

—北京：清华大学出版社，2002

书名原文：Definitive XML Schema

ISBN 7-302-06096-7

I. X... II. ①沃...②陈...③乔...④英... III. 可扩充语言，XML—程序设计 IV.TP312

中国版本图书馆 CIP 数据核字(2002)第 091354 号

出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)

<http://www.tup.tsinghua.edu.cn>

责任编辑：陈宗斌

封面设计：康博

版式设计：康博

印 刷 者：北京通州区大中印刷厂

发 行 者：新华书店总店北京发行所

开 本：787×1092 1/16 印张：19.5 字数：499 千字

版 次：2003 年 1 月第 1 版 2003 年 1 月第 1 次印刷

书 号：ISBN 7-302-06096-7/TP·3641

印 数：0001~3000

定 价：46.00 元

目 录

第 1 章 XML Schema 简介	1
1.1 XML Schema 的概念	1
1.2 模式的用途	2
1.2.1 数据确认	2
1.2.2 交易双方的合约	2
1.2.3 系统文档	2
1.2.4 数据扩充	2
1.2.5 应用程序信息	2
1.3 模式设计	3
1.3.1 准确性和精确性	3
1.3.2 明晰性	3
1.3.3 广泛适用性	3
1.4 模式语言	4
1.4.1 文档类型定义	4
1.4.2 进入模式	4
1.4.3 W3C 的 XML Schema	5
1.4.4 术语说明	5
1.4.5 其他模式语言	5
第 2 章 XML Schema 快速入门	8
2.1 示例模式	8
2.2 XML Schema 的组件	9
2.2.1 声明与定义的比较	9
2.2.2 全局组件与局部组件的比较	9
2.3 元素和属性	9
2.4 数据类型	10
2.4.1 简单类型和复杂类型	10
2.4.2 命名类型和匿名类型	11
2.4.3 类型定义层次	11
2.5 简单类型	11
2.5.1 内置简单类型	11
2.5.2 限制简单类型	12
2.5.3 列表类型和联合类型	12



2.6	复杂类型	13
2.6.1	内容类型	13
2.6.2	内容模型	13
2.6.3	派生复杂类型	14
2.7	命名空间和 XML Schema	15
2.8	模式组成	16
2.9	实例和模式	16
2.10	注解	17
2.11	高级特性	17
2.11.1	可重用组	17
2.11.2	一致性约束	18
2.11.3	替代组	18
2.11.4	重定义	18
第 3 章	命名空间	19
3.1	XML 中的命名空间	19
3.1.1	命名空间名称是 URI	19
3.1.2	命名空间声明和前缀	19
3.1.3	默认命名空间声明	21
3.1.4	名称术语	21
3.1.5	命名空间声明的作用域	22
3.1.6	重写命名空间声明	22
3.1.7	属性和命名空间	23
3.1.8	总结示例	24
3.2	命名空间和模式的关系	25
3.3	在 XSDL 中使用命名空间	26
3.3.1	目标命名空间	26
3.3.2	XML Schema 命名空间	27
3.3.3	XML Schema 实例命名空间	27
3.3.4	模式文档中的命名空间声明	27
第 4 章	模式组成	30
4.1	模块化模式文档	30
4.2	定义模式文档	31
4.3	模式组合	32
4.3.1	组合多个文档的模式	32
4.3.2	限定名称的惟一性	33
4.3.3	缺少的组件	33
4.3.4	模式文档默认值	34

4.4	include、redefine 和 import	34
4.4.1	包含(include)	34
4.4.2	重定义(redefine)	37
4.4.3	导入(import)	37
第 5 章	实例和模式	40
5.1	使用实例属性	40
5.2	模式处理	41
5.2.1	验证	41
5.2.2	扩充实例	41
5.3	使实例与模式相关	42
5.4	在实例中使用 XSDL 提示	42
5.4.1	xsi:schemaLocation 属性	42
5.4.2	xsi:noNamespaceSchemaLocation 属性	43
5.5	间接引用命名空间	44
5.6	根元素	45
5.7	协同使用 DTD 和模式	47
5.8	使用特定模式处理器	48
5.8.1	XSV	48
5.8.2	Xerces	49
5.8.3	Oracle XDK	49
5.8.4	Microsoft MSXML	50
第 6 章	模式文档和扩展	52
6.1	机制	52
6.1.1	注解	52
6.1.2	用户文档	53
6.1.3	应用程序信息	54
6.1.4	验证注解	54
6.1.5	非原属性	55
6.1.6	设计提示: 应该使用注解还是非原属性	56
6.2	用户文档	56
6.2.1	用户文档类型	56
6.2.2	数据元素定义	57
6.2.3	代码文档	57
6.2.4	分段备注	57
6.3	应用程序信息	59
6.3.1	应用程序信息的类型	59
6.3.2	同现约束的 Schematron	59



6.3.3	RDBMS 映射的模式附属框架	60
6.4	符号	61
6.4.1	声明符号	62
6.4.2	声明符号属性	62
6.4.3	符号和未解析的实体	64
第 7 章	元素声明	65
7.1	全局和局部元素声明	65
7.1.1	全局元素声明	65
7.1.2	局部元素声明	67
7.1.3	设计提示：应该使用全局还是局部元素声明	68
7.2	声明元素的数据类型	69
7.3	默认值和固定值	70
7.3.1	默认值	70
7.3.2	固定值	71
7.4	零值和置零性	72
7.4.1	在实例中使用 <code>xsi:nil</code>	74
7.4.2	使元素可置零	75
7.5	限定与非限定形式	75
第 8 章	属性声明	76
8.1	全局和局部属性声明	76
8.1.1	设计提示：应该使用属性还是元素	76
8.1.2	全局属性声明	77
8.1.3	局部属性声明	78
8.1.4	设计提示：应该在全局还是局部声明属性	78
8.2	为属性指派类型	79
8.3	默认值和固定值	80
8.3.1	默认值	80
8.3.2	固定值	80
8.4	限定与非限定形式的比较	81
第 9 章	简单类型	82
9.1	简单类型的种类	82
9.2	简单类型的定义	83
9.2.1	命名简单类型	83
9.2.2	匿名简单类型	84
9.2.3	设计提示：应该使用命名类型还是匿名类型	84
9.3	简单类型的限制	85

9.3.1 定义限制	86
9.3.2 面的综述	86
9.3.3 继承与限制面	87
9.3.4 固定面	89
9.4 面	89
9.4.1 界限面	89
9.4.2 长度面	90
9.4.3 totalDigits 和 fractionDigits	91
9.4.4 枚举	91
9.4.5 样式	93
9.4.6 Whitespace	94
9.5 阻止简单类型派生	95
第 10 章 正则表达式	97
10.1 正则表达式的结构	97
10.2 基本单元	98
10.2.1 标准字符	98
10.2.2 字符类换码	99
10.2.3 字符类表达式	105
10.2.4 加括号的正则表达式	107
10.3 量词	108
第 11 章 联合类型与列表类型	109
11.1 种类和派生类型	109
11.2 联合类型	110
11.2.1 定义联合类型	110
11.2.2 限制联合类型	111
11.2.3 联合的联合	112
11.2.4 在实例中指定成员类型	112
11.3 列表类型	113
11.3.1 定义列表类型	113
11.3.2 设计提示：应该何时使用列表	114
11.3.3 限制列表类型	115
11.3.4 列表与字符串	117
11.3.5 联合的列表	118
11.3.6 列表的列表	119
11.3.7 限制项目类型	119



第 12 章 内置简单类型	120
12.1 内置类型	120
12.2 基于字符串的类型	121
12.2.1 string、normalizedString 与 token	121
12.2.2 Name	123
12.2.3 NCName	124
12.2.4 language	124
12.3 数字类型	126
12.3.1 浮点与双精度	126
12.3.2 小数	127
12.3.3 整数类型	127
12.4 日期和时间类型	129
12.4.1 date	129
12.4.2 time	130
12.4.3 dateTime	131
12.4.4 gYear	131
12.4.5 gYearMonth	132
12.4.6 gMonth	132
12.4.7 gMonthDay	133
12.4.8 gDay	133
12.4.9 duration	134
12.4.10 表示时区	135
12.4.11 面	135
12.4.12 日期和时间排序	135
12.5 继承类型	136
12.5.1 ID	136
12.5.2 IDREF	137
12.5.3 IDREFS	138
12.5.4 ENTITY	138
12.5.5 ENTITIES	139
12.5.6 NMTOKEN	140
12.5.7 NMTOKENS	141
12.5.8 NOTATION	142
12.6 其他类型	142
12.6.1 QName	142
12.6.2 boolean	143
12.6.3 hexBinary 和 base64Binary	143
12.6.4 anyURI	144

12.7	类型等同性	145
第 13 章	复杂类型	147
13.1	复杂类型的概念	147
13.2	定义复杂类型	147
13.2.1	命名复杂类型	147
13.2.2	匿名复杂类型	149
13.2.3	复杂类型选择	149
13.3	内容类型	150
13.3.1	简单内容	150
13.3.2	纯元素内容	150
13.3.3	混合内容	151
13.3.4	空内容	152
13.4	使用元素类型	152
13.4.1	局部元素声明	152
13.4.2	元素引用	152
13.4.3	元素通配符	154
13.4.4	元素类型名称的重复	156
13.5	使用模型组	157
13.5.1	sequence 组	157
13.5.2	choice 组	159
13.5.3	sequence 和 choice 组的嵌套	160
13.5.4	all 组	161
13.5.5	命名模型组引用	162
13.5.6	确定性内容模型	163
13.6	使用属性	164
13.6.1	局部属性声明	164
13.6.2	属性引用	164
13.6.3	属性通配符	166
13.6.4	属性组引用	167
第 14 章	派生复杂类型	168
14.1	派生类型的作用	168
14.2	限制和扩展	168
14.3	简单内容和复杂内容	169
14.3.1	simpleContent 元素	169
14.3.2	complexContent 元素	169
14.4	复杂类型扩展	170
14.4.1	简单内容扩展	170



14.4.2	复杂内容扩展	171
14.4.3	混合内容扩展	173
14.4.4	空内容扩展	174
14.4.5	属性扩展	174
14.4.6	属性通配符扩展	175
14.5	复杂类型限制	176
14.5.1	简单内容限制	177
14.5.2	复杂内容限制	178
14.5.3	混合内容限制	185
14.5.4	空内容限制	186
14.5.5	属性限制	187
14.5.6	属性通配符限制	189
14.6	类型替代	190
14.7	控制类型派生和替代	191
14.7.1	final: 阻止复杂类型派生	191
14.7.2	block: 阻止派生类型的替代	192
14.7.3	阻止元素声明中的类型替代	193
14.7.4	abstract: 强制派生	193
第 15 章	可重用组	195
15.1	可重用组的作用	195
15.2	命名模型组	195
15.2.1	定义命名模型组	195
15.2.2	引用命名模型组	197
15.3	属性组	199
15.3.1	定义属性组	200
15.3.2	引用属性组	201
15.4	可重用组与复杂类型派生	204
第 16 章	替代组	206
16.1	替代组的作用	206
16.2	替代组的层次结构	206
16.3	声明替代组	207
16.4	替代组的类型约束	209
16.5	替代组的替换	210
16.5.1	可重用 choice 组	210
16.5.2	在实例中替代派生类型	211
16.6	控制替代组	212
16.6.1	final: 阻止替代组声明	213

16.6.2	block: 在实例中阻止替代	213
16.6.3	abstract: 强制替代	214
第 17 章	一致性约束	215
17.1	一致性约束类别	215
17.2	设计提示: 应该使用 ID/IDREF 还是 key/keyref	215
17.3	一致性约束的结构	215
17.4	唯一性约束	217
17.5	关键字约束	218
17.6	关键字引用	219
17.7	选择器与字段	221
17.7.1	选择器	221
17.7.2	字段	222
17.8	XML Schema 的 XPath 子集	223
17.9	一致性约束和命名空间	224
第 18 章	重定义模式组件	227
18.1	重定义基础	227
18.1.1	包含和重定义	228
18.1.2	重定义和命名空间	228
18.1.3	扩大的影响	228
18.2	重定义机制	228
18.3	重定义简单类型	229
18.4	重定义复杂类型	230
18.5	重定义命名模型组	231
18.5.1	定义子集	231
18.5.2	定义超集	232
18.6	重定义属性组	233
18.6.1	定义子集	233
18.6.2	定义超集	234
第 19 章	关于 DTD	236
19.1	元素声明	236
19.1.1	简单类型	236
19.1.2	带有简单内容的复杂类型	237
19.1.3	带有复杂内容的复杂类型	237
19.1.4	混合内容	239
19.1.5	空内容	239
19.1.6	任何内容	240



19.2	属性声明	240
19.2.1	属性类型	240
19.2.2	枚举属性类型	241
19.2.3	Notation 属性	241
19.2.4	默认值	242
19.3	符号	243
19.4	可重用的参数实体	243
19.4.1	重用内容模型	243
19.4.2	重用属性	244
19.5	用于可扩展性的参数实体	245
19.5.1	sequence 组的扩展	245
19.5.2	choice 组的扩展	246
19.5.3	属性扩展	247
19.5.4	属性组扩展	248
19.6	外部参数实体	249
19.7	通用实体	250
19.7.1	字符和其他已分析的实体	250
19.7.2	未分析的实体	250
19.8	注释	251
19.9	协同使用 DTD 和模式	252
第 20 章	命名考虑事项	253
20.1	命名指导原则	253
20.1.1	有效 XML 名称的规则	253
20.1.2	分隔符	253
20.1.3	名称长度	254
20.1.4	标准术语和缩写	254
20.1.5	主题术语的使用	254
20.2	限定与非限定名称	255
20.2.1	限定的局部名称	256
20.2.2	非限定的局部名称	256
20.2.3	使用 elementFormDefault	256
20.2.4	形式和全局元素声明	258
20.2.5	默认命名空间和非限定名称	258
20.2.6	设计提示：应该使用限定的还是非限定的局部名称	258
20.2.7	限定的与非限定的属性名称	259
20.3	构造命名空间	260
20.3.1	同一个命名空间	260

20.3.2	不同的命名空间	262
20.3.3	可变命名空间	265
20.4	多种语言	267
第 21 章	可扩展性和重用	270
21.1	重用	270
21.2	扩展模式	271
21.2.1	通配符	271
21.2.2	类型派生	273
21.2.3	替代组	274
21.2.4	类型重定义	276
21.2.5	命名模型组重定义	277
21.3	模式的版本管理	278
21.3.1	模式兼容性	278
21.3.2	应用程序兼容性	279
21.3.3	转换功能	279
21.3.4	使用版本号	279
21.4	设计支持变化的应用程序	280
附录 A	XSDL 关键字表	281
A.1	XSDL 元素类型	281
A.2	XSDL 属性	287
附录 B	内置简单类型	293
B.1	内置简单类型	293
B.2	对于内置简单类型的适用性	295

第1章 XML Schema简介

本章简要地介绍了模式(Schema)及其重要意义。同时也讨论了基本的模式设计目标，并介绍了现有的多种模式语言。

1.1 XML Schema 的概念

模式(schema)这个词语表示图解、计划或框架。在 XML 中，它指描述 XML 文档的文档。假使您有一个例 1-1 所示的 XML 实例，它包含了一个 product 元素和一个属性(effDate)，前者又有两个子元素(number 和 size)。

例 1-2 给出了一个描述这个实例的模式。该模式中包含了元素和属性声明，用于将数据类型和元素类型名赋予元素和属性。

例子 1-1 Product 实例

```
<product effDate="2001-04-02">
  <number>557</number>
  <size>10</size>
</product>
```

例子 1-2 Product 模式

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="product" type="ProductType"/>
  <xsd:complexType name="ProductType">
    <xsd:sequence>
      <xsd:element name="number" type="xsd:integer"/>
      <xsd:element name="size" type="SizeType"/>
    </xsd:sequence>
    <xsd:attribute name="effDate" type="xsd:date"/>
  </xsd:complexType>
  <xsd:simpleType name="SizeType">
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="2"/>
      <xsd:maxInclusive value="18"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```



1.2 模式的用途

1.2.1 数据确认

模式最常见的用法之一就是依照预定义好的规则来验证 XML 文档的有效性。模式可以确认：

- 元素和属性的结构。例如，一个 `product` 对象必须有 `number` 和 `size`，以及可选的 `effDate`（有效日期）。
- 元素的顺序。例如 `number` 必须出现在 `size` 之前。
- 元素和属性的数据值，根据范围、枚举以及样式匹配。例如，`size` 必须为一个 2 到 18 间的整数，而 `effDate` 必须是有效的日期。
- 实例中值的惟一性。例如，实例中所有的产品编号必须是惟一的。

1.2.2 交易双方的合约

通常情况下，XML 实例会在组织之间传递。模式就如同交易双方都必须遵循的合约。它清楚地说明了文档结构的规则以及要求。如果一个实例被模式验证过，那么“合约”就可以使用可用的工具来实施。

1.2.3 系统文档

模式可以为 XML 实例中的数据提供文档。任何需要了解这些数据的人，都可以从模式中获得关于项目的名称、结构和数据类型等信息。要包括附加文档，您也可以为模式中的组件添加注解。

1.2.4 数据扩充

模式处理过程也可以添加到实例中。它为元素和属性插入默认和固定的值，并根据数据类型规范化空白空间。

1.2.5 应用程序信息

应用程序在处理文档中的一些特殊类型时，模式提供了一种方式来为应用程序提供数据的附加信息。例如，您可以包含一些关于如何将 `product` 元素实例映射到数据库表的信息，并且允许应用程序使用这些信息来自动更新带有该数据的特定的表。

除了处理时可用之外，模式中的该信息也可以用于生成代码，比如：

- 用来修改该信息的用户界面。例如，如果已经知道 `size` 是介于 2 到 18 之间的数值，那么可以生成一个带有滑动条的界面，以这两个数作为滑动条的限制值。
- 将实例数据转换为如 XHTML 这样容易阅读的表现形式的样式表。例如，如果您知道 `number` 元素的内容有一个易于理解的名称为“Product Number”，那么就可以将它作为列标题。

- 从数据库中插入或提取数据的代码。例如，如果您知道产品编号对应于 **PRODUCTS** 表中的 **PROD_NUM** 列，那么您能够生成一个有效的例程来将它插入到该列。

一些工具已经开始利用模式的这种能力。在未来数年中，人们将以许多有创造性的新方法使用模式。

1.3 模式设计

XML Schema 功能多样，经常可以通过不同的方法来精确地描述同一事物。在设计模式时选择不同的方法可能影响到模式的可用性、准确性和适用性。因此，最重要的就是在创建模式时，要明确地知道自己的设计目标。这些目标可能会因为使用 XML 的方法不同而不同，但有些目标对于各种应用案例来说是通用的。

1.3.1 准确性和精确性

显而易见，模式应该能准确地描述 XML 实例，并且能够确保该实例的有效性。模式在描述数据时也应该是精确的。精确性能带来更完整的有效性，以及结构更好的文档。精确性可以通过定义真实代表有效值的受限数据类型来实现。

1.3.2 明晰性

模式应该非常清晰，允许读者直观地理解被描述实例的结构和特征。模式的明晰性可以通过以下方式实现：

- 选择合适的名称
- 命名规则的一致性
- 结构的一致性
- 良好的文档
- 避免不必要的复杂性

1.3.3 广泛适用性

仅仅是因为一些特殊的应用目的而创建模式吗？有时候，这可能是适用的。但最好使创建的模式有更广泛的适用性。例如，一家商业公司如果只处理国内账目，那么可能就不会将 **country** 元素声明作为 **address** 的一部分。出于一致性和将来应用的目的，他们应该考虑将它作为一个可选的元素添加进去。

模式的广泛适用性包含了两部分：可重用性和可扩展性。可重用的模式组件是模块化的和格式良好的，鼓励模式编写者在其他模式中重用它们。可扩展的组件则是灵活的和开放的，允许其他模式编写者在原有基础上根据需求进行扩展以备将来之用。由于可重用性和可扩展性非常重要，本书第 21 章的全部内容都是对它们的详细讲述。