ICS 35.240.30
L 74

0800318

中华人民共和国国家标准化指导性技术文件

GB/Z 21025—2007

XML 使用指南

XML user's guide

华南理工大学
图书馆藏书

2007-06-29 发布

数码防伪

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会 发布

中 华 人 民 共 和 国
国家标准化指导性技术文件
XML 使用指南
GB/Z 21025—2007
*
中国标准出版社出版发行
北京复兴门外三里河北街 16 号
邮政编码:100045
网址 www.spc.net.cn
电话:68523946 68517548
中国标准出版社秦皇岛印刷厂印刷
各地新华书店经销
*
开本 880×1230 1/16 印张 3.25 字数 92 千字
2007 年 11 月第一版 2007 年 11 月第一次印刷
*
书号:155066·1-30020 定价 34.00 元

如有印装差错 由本社发行中心调换
版权专有 侵权必究
举报电话:(010)68533533

GB/Z 21025-2007

前　言

本指导性技术文件仅供参考。有关对本指导性技术文件的建议和意见，请向国务院标准化行政主管部门反映。

本指导性技术文件的附录A和附录B为资料性附录。

本指导性技术文件由中华人民共和国信息产业部提出。

本指导性技术文件由中国电子技术标准化研究所归口。

本指导性技术文件起草单位：中国电子技术标准化研究所、北京信息工程学院、万达信息技术公司、北京航天航空大学、方正电子技术公司。

本指导性技术文件主要起草人：李宁、顾晓毅、林学练、王国印、吴志刚、赵菁华。

引　言

本指导性技术文件规定了使用可扩展置标语言(XML)应该遵循的原则和注意事项,适用于XML应用的开发者、管理者、使用者和其他关心XML使用的人员。

在GB/T 18793—2002《信息技术　可扩展置标语言(XML)1.0》中,描述了可扩展置标语言(eXtensible Markup Language,XML)。它是标准通用置标语言(Standard Generic Markup Language,SGML)的一个子集,其目的是使SGML文档在web应用中可以像HTML文档一样进行发送、接收和处理。为此,XML的设计力求易于实现,并能与SGML和HTML很好地互操作。它针对web应用,简化了一些SGML的不常用的内容。

GB/T 18793—2002所定义的XML是一组用来构造语义置标的规则集合,通过这些置标文档的各个部分可按预先定义的语义结构组织起来并进行结构化验证。GB/T 18793—2002的重点在于对以DTD为核心的XML本身的语法进行描述,并不包括Schema的内容,具体的使用方法在那里也少有涉及,这些将在本指导性技术文件中做出解释,以帮助XML的推广使用。本指导性技术文件还给出了部分使用XML的例子。本指导性技术文件为设计一致性的、合理的DTD和Schema提供了一个指导性框架,将有助于更好地理解XML,重用XML组件,并达到良好的互操作性。

本指导性技术文件的重点在于如何采用元语言标准和基础标准来定义应用标准,核心是行业应用词汇表的设计。在本指导性技术文件中,部分内容仅适用于Schema或仅适用于DTD,将特别标出。未被特殊标出的部分两者都适合。另外为了查阅方便,本指导性技术文件的大部分条目标题直接采用了原则性陈述的形式。

目　　次

此为试读，需要完整PDF请访问：www.ertongbook.com

XML 使用指南

1 范围

本指导性技术文件给出了指导 XML 文档的编写的原则，包括标准的符合性原则，国际化和本地化原则，组件命名原则，命名空间使用原则，词汇表编写原则，类型、元素与属性的使用原则，版本与注释的使用原则，实例的编写原则，XML 解析器及选择，应用开发过程和注册规程等内容。

本指导性技术文件适用于 XML 的各类开发人员和使用人员。

2 规范性引用文件

下列文件中的条款通过本指导性技术文件的引用而成为本指导性技术文件的条款。凡是注日期的引用文件，其随后所有的修改单(不包括勘误的内容)或修订版均不适用于本指导性技术文件，然而，鼓励根据本指导性技术文件达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件，其最新版本适用于本指导性技术文件。

GB/T 1988—1998 信息技术 信息交换用七位编码字符集(eqv ISO 646:1991)

GB 2312—1980 信息交换用汉字编码字符集 基本集

GB/T 13000.1 信息技术 通用多八位编码字符集(UCS) 第一部分：体系结构与基本多文种平面(GB/T 13000.1—1993，idt ISO/IEC 10646-1:1993)

GB/T 14814—1993 信息技术 文本和办公系统 标准通用置标语言(SGML)(idt ISO 8879:1986)

GB 18030—2000 信息技术 信息交换用汉字编码字符集 基本集的扩充

GB/T 18391(所有部分) 信息技术 数据元的规范和标准化

GB/T 18793—2002 信息技术 可扩展置标语言(XML)1.0(neq W3C RFC-xml-19980210:1998)

3 术语和定义

GB/T 18793—2002 中确定的以及下列术语和定义适用于本指导性技术文件。

3.1

文档对象模型 Document Object Model:DOM

W3C 制定的 XML 应用程序接口。它将 XML 文档表示成一个树形的结构。DOM 规定了一系列编程指令，允许应用程序多次访问并操作文档树的组件。

3.2

元数据 metadata

定义和描述其他数据的数据。

3.3

资源目录描述语言 Resource Directory Description Language:RDDL

一种符合 XHTML 格式的资源目录和描述，用于为用户提供目标资源的相关信息。

3.4

XHTML

W3C 为 HTML 制定的 XML 词汇表。

3.5

XML 的简单 API　Sample API for XML;SAX

为序列化存取 XML 文档信息所制定的一种接口。SAX 处理器在处理 XML 文档的时候，随着遇见开、关标记和字符数据等内容，不断触发事件，调用应用程序(事件处理程序)处理，能够达到比较快的速度和效率。

3.6

式样单　stylesheet

一套指令集合，主要用于规定 XML 文档显现格式，也可以将一个 XML 文档转换成另一个文档。

3.7

XML 组件　XML component

XML 元素、元素属性和 XML 词汇表的统称。

3.8

XML　schema/Schema

一种用于限定文档结构(如元素的顺序、出现次数、属性等)的机制，用于描述一类实例文档的结构。解析器可以根据 schema 来验证文档。本指导性技术文件中，用小写字母开头的 schema 统称这一概念，其中也包括 DTD。用大写字母开头的 Schema 特指 W3C 制定的 Schema 标准(REC-xmlschema-0-20010502 Part 0～2)。

3.9

XML 命名空间　XML namespace

为了解决命名冲突，为元素和属性命名引入的逻辑空间，是在 XML 文档中通过 URI 引用声明的，并采用限定性前缀将元素和属性与命名空间联系起来。

3.10

XML 词汇表　XML vocabulary

在特定领域给特定用户群使用、有确定的功能的数据元集合以及数据模型，是表示一类文件的结构的 schema 的统称。

3.11

XML 作品　XML artifacts

具有独立保存价值的各种 XML 数据，包括 schema、式样单以及 XML 实例文档等。

3.12

可扩展式样语言　eXtensible Stylesheet Language;XSL

由 W3C 组织制定的用于定义 XML 文档转换和显现的系列标准，包括：XSLT、XPath、XSL-FO 三部分。

3.13

可扩展式样语言转换　eXtensible Stylesheet Language Transformations;XSLT

由 W3C 组织制定的用于转换 XML 文档的语言。

3.14

统一建模语言　Unified Modeling Language;UML

为创建商业和技术模型定义的一种语言和图形表示法，它定义了多种模型种类，涵盖了从功能需求定义、事务处理活动工作流模型到逻辑和物理层次等软件开发的各个方面。

3.15

上驼峰形式大小写　Upper Camel Case;UCC

把单词拼接在一起的一种方式，不使用含下划线“_”和句点“.”等的连字符，每个单词的首字母大写，其余字母均小写。在有大写缩写字母或数字的时候，后面的单词首字母小写。

例如：NameUsedInXMLschema。

3.16

下驼峰形式大小写 Lower Camel Case；LCC

LCC 与 UCC 的唯一区别是整个名称的首字母用小写，例如：attributeValue。

3.17

统一建模方法 Unified Modeling Methodology；UMM

UN/CEFACT 为事务处理建模，支持下一代电子数据交换（Electronic Data Interchange，EDI）开发而推出的一套建议的方法。它基于 Rational 统一过程理论，使用 UML 作为建模语言。

3.18

Unicode

由 Unicode 协会（Unicode consotium）制定的通用字符。其主要目的是为纯文本内容提供一套无歧义的编码，以方便全球各种语言文字的转换。

3.19

统一资源定位符/统一资源指示符/统一资源名称 URL（Uniform Resource Locators）/URI（Uniform Resource Indicators）/URN（Uniform Resource Names）

网络环境下三种不同的，但又相关的引用资源的统一的方法。

4 标准的符合性原则

所有 XML 文档的编制和所有的处理工具或处理器，包括解析器、文档生成器、验证工具，以及所有使用 XML 的软件应符合 GB/T 18793—2002，同时也应该考虑符合国际权威标准化组织订立的其他正式标准。对于标准未涉及到的，而应用中需要的对 XML 的扩充，原则上仅限于在组织内部使用，不宜公开传播，除非经过正式的审查和注册。

5 国际化和本地化原则

5.1 XML 文档编码

GB/T 18793—2002 规定，XML 文档可以使用以下编码字符集：

——GB 18030—2000；

——GB 13000.1；

——GB 2312—1980；

——其他 XML 处理器支持的编码字符集。

在 GB/T 18793—2002 中，缺省字符集规定为 GB 13000.1，亦称为通用字符集（Universal Character Set，UCS）。

注：Unicode 是由 Unicode 协会（Unicode Consotium）制定的通用字符集。其主要目的是为纯文本内容提供一套无歧义的编码，以方便全球各种语言文字的转换。在 W3C 的 XML 1.0 中，大量使用了 Unicode。其 2004 年发布的 XML 1.1 中作了更新，使 XML 不再依赖于 Unicode 的特定版本。由于 GB/T 13000.1—1993 与 Unicode（2.0 版本以上）是完全兼容的，本指南中除非特殊需要不涉及 Unicode。

在 XML 文档交换中经常用到 UCS-4、UCS-2 以及 UTF-8、UTF-16 等编码形式，简要介绍如下。

UCS 的双八位的 BMP 形式（UCS-2）规定每个字符用两个字节编码，这种形式仅适用于基本多语种平面。如“一”的双八位形式为 4E00。

UCS 的肆八位的正则形式（UCS-4）规定每个字符用四个字节编码，例如：汉字“一”的正则形式为 0000 4E00。

在 UCS 中，编码点在 0～65535 的字符归属第 0 平面，也称基本多语种平面（Basic Multilingual Plane，BMP）。这个平面中包含大部分全世界正在使用的公用字符，包括来自罗马字母、西里尔字母、阿

拉伯语、希腊语、希伯来语、常用汉字和其他语言的文字。编码点在65536～131071的字符归属第1平面。这个平面包括音乐符号、数学符号和一些已经不再使用的语言(如古意大利语)的文字。编码点在131072～196607的字符归属第2平面,收录了很多不常用汉字。第14平面包含了一些语言标记,因为XML有xml:lang属性可用而完全不需理会这些标记。其他的平面至今都没有很好地定义。

在实际使用中更多采用的是UTF-8和UTF-16。制定UTF-8的目的是为了与原8比特系统向下兼容;制定UTF-16的目的是为了向上发展和扩充。

UTF-8(Unicode Transformation Format, 8-bit encoding form)是一种变长编码。编码点为0～127的每一个字符(GB/T 1988—1998字符)占一个字节,编码点为128～4095的每一个字符占据两个字节,第0平面的其他字符每一个占据3个字节,从第1至第15平面的每一个字符占据4个字节。UTF-8有很多优点,列举如下:

——它是GB/T 1988—1998的超集,因此对于纯英文的文本,一个UTF-8文件与GB/T 1988—1998文件完全一样,非常利于兼容,因此XML把UTF-8选作缺省的编码形式。

——所有的GB/T 1988—1998字符都不会成为其他字符编码的一部分,因此非常容易分辨GB/T 1988—1998字符。

——UTF-8与字节顺序无关。在计算机系统中,大数值类型(如整型)使用多个字节表示,不同体系结构采用的字节排列顺序不同。其中,部分采用由高字节到低字节的排列顺序,称为big-endian;其他则采用由低字节到高字节的排列顺序,称little-endian。对于大多数big-endian的UNIX系统和little-endian的Windows系统,对同一个文档UTF-8可以做到每个字符一一对应,因此,没有必要在XML文档开头放置字节顺序标记。

——从单一字节就可以判断字符边界。只观察单一字节,程序就可以判断该字节是下列哪种情形之一:单字节字符、双字节字符的第一个字节,三字节字符的第一、二、三字节。

——对于常见字符组成的文档,UTF-8占用空间最节省。

UTF-16(Unicode Transformation Format, 16-bit encoding form)也是一种变长编码。在UTF-16中,编码点为0～65535的字符使用单一的16位编码单元表示;而编码点为65536～1114111的字符使用一对16位编码单元表示(RC-element或surrogate pair)。UTF-16最大的好处是优化了基本多语种平面的字符表示,每个字符只需要2个字节,可作为定长编码来有效使用。对于大量使用中、日、韩文字的文本,其占用空间比UTF-8约节省1/3。然而比起UTF-8,UTF-16丧失了很多优点,最主要的是UTF-16是字节顺序相关的,为解决字节顺序问题,要在XML文档开头加一个字节顺序标志(＃xFEFF)。如果程序读出的是FE和FF,则可以断定文档的编码是big-endian UTF-16;如果读出的是FF和FE,则文档的编码是little-endian UTF-16。由于＃xFEFF不是一个合法的GB/T 13000字符,所以不会与其他内容混淆。UTF-16的另一个缺点是检测字符边界比较麻烦。

除了UTF-8和UTF-16,GB/T 13000.1还有UTF-32编码形式,与UCS-4一致。然而常用的只有UTF-8和UTF-16。一般原则是,如果文档不含大量的中、日、韩文字,XML应该使用UTF-8作为缺省的编码,否则应该使用UTF-16。如果难以判断,仍可采用UTF-8。

但是,除非特别必要,XML置标标记(包括元素名和属性名)应该尽量采用GB/T 1988—1998字符集,以适应当前很多工具对GB 13000.1和其他编码字符集支持不够完善的现况。尽量不要使用GB 13000.1以外的字符集,如果确实必要应该考虑采用编码转换,将其他字符集编码映射到GB 13000.1再处理。

注:在使用GB 13000的时候,应该遵照XML 1.1标准的建议先将XML文本规格化。因为在GB 13000中,一些文本成分即可使用静态的预先组合好的形式,也可使用动态组合的形式。例如"é"可以表达为单个字符"＃xE9",也可以表达为两个字符连用,即"＃x65"和"＃x301"。为了进行字符比较,需要进行规格化,即使用一种规范化的、单一的GB/T 13000文本形式来表示这些成分。Unicode定义了四种规范化形式:Normalization Form D (NFD),Normalization Form KD (NFKD),Normalization Form C (NFC)和Normalization Form KC(NFKC)。

其中 NFD 和 NFKD 将可能的字符进行分解，而 NFC 和 NFKC 将可能的字符进行组合。XML 1.1 规定规范化的文本应该采用 NFC 的形式。

5.2 URI 字符集使用

统一资源指示器(URI)用于定位系统中的某项资源。统一资源定位器 (URL)(较 URI 更为人所熟知)是 URI 的子集。XML 通常使用 URI 指定链接中的资源，定义命名空间等。以往的 URI 基于 GB/T 1988—1998 的基本集，无法在 URI 中直接表示扩展字符，但可以通过转义机制来进行。

对于 URI 中的保留字符、空白字符以及其他不满足条件或非安全的字符(包括 GB/T 1988—1998 中编码点 127 之后的字符、空白符、控制符，如“{”、“}”、“|”、“、”、“～”、“[”、“]”等)都应该编码成“%hh”形式，其中，hh 是该字符在字符集中的 16 进制的编号。

以下是一个尚未编码的 URL 例子：

```
http://www.company.com/中国
```

转换后成为：

```
http://www.company.com/%E4%B8%AD%E5%9B%BD
```

W3C 和 IETF 正在制定基于 GB/T 13000 的国际化的 URI 标准。在正式标准发布之前，仍然需要转义非安全的字符。

5.3 谨慎使用多语种文档

多语种文档有两类，一类是在内容级别上使用多种语言，即元素内容使用不同的语言，另一类是在结构级别上使用多种语言，即用不同语言表示一个元素。多语言文档不易处理，特别是后者，不易保持几种语言版本的一致性，不易为检索等应用建立等价元素关系，因而需要尽力避免使用多语言文档。如果确实需要，应该尽量采用工具进行转换，并通过标识符建立元素之间的关联。

5.4 XML 中多语种的使用

除非在其他标准、规范或既定应用中明确规定了须使用其他语种，在 XML 的以下成分中可以使用中文：

——元素名和元素内容；
——属性名和属性值；
——枚举值；
——唯一标识符；
——记法(Notation)；
——文本字符串。

面向公共使用的词汇表应该首先准备一个语种的主版本。为了利于使用其他语种的地区和国家使用，可以将该词汇表主版本翻译成其他语种的副版本，例如：由英文主版本翻译成中文或汉语拼音版本。建议通过一套 XSL 式样单(stylesheet)或程序进行词汇表之间的转换。经过转换的词汇表应该通过注释指明原始词汇表的出处。另外，在词汇表的不同版本中，应该使用标识符(建议使用“locID”)来标识等价的元素，并使用特定的属性(建议使用“attrList”)来标识属性顺序，以利在一个词汇表的不同语种实例之间能仍然保持严格的元素和属性的语义关联，并方便 XML 检索等处理机制识别。这方面的一个实例见 14.1。

[适用于 Schema] 某些时候，也可采用元素替换(substitutionGroup)的方式，在一套 Schema 中指定多个语种的等价元素或属性名称。例如：

```
〈xsd:element name="Subway" type="xsd:string"/〉
〈xsd:element name="Metro" substitutionGroup="Subway" type="xsd:string"/〉
〈xsd:element name="地铁" substitutionGroup="Subway" type="xsd:string"/〉
```

这种方法要求 Schema 的编码方法可以编码各个所需的语种，而且 XML 处理器可以处理这样的编码。另外，substitutionGroup 中的替换元素必须是全局的。它的优点是能够保证多个语种文档的一致性。

通过 Schema 的命名空间可以建立多个版本的联系，例如：

```
〈xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                    targetNamespace="http://www.books.org/zh-cn-py"
                    xmlns=http://www.books.org/zh-cn-py〉
```

在 DTD 中，可以通过前导说明部分的 DOCTYPE 声明来建立联系：

```
〈! DOCTYPE Book PUBLIC "universal/Publishing/Book/ZH-CN-PY"〉
```

在进行词汇表语种转换的过程中，要注意妥善处理好命名空间的对应关系。

XML 注册机构最好能提供查询机制以方便检索同一词汇表的不同编码版本。

5.5 使用 xml:lang 整合语种声明

明确指示文档内容（如时间、日期、货币符号和数字）所使用的语言，可以有助于采用适合的显现式样，或正确地解释与语种无关的数据。所有与语种相关的元素或属性内容，均应使用该属性进行声明。

对于 schema 中使用多语种元素的情况，应该为所有与方言相关的元素指定 xml:lang 属性。例如：

```
〈ProductName id="id1"〉
  〈var xml:lang="en"〉disc〈/var〉
  〈var xml:lang="zh-cn"〉光盘〈/var〉
  〈var xml:lang="zh-tw"〉光碟〈/var〉
〈/ProductName〉
```

xml:lang 所使用的语种代码，与中文相关的有如下几个：

——zh　　中文（简体）；

——zh-cn　中文-中国（简体）；

——zh-hk　中文-香港（繁体）；

——zh-mo　中文-澳门（简体）；

——zh-sg　中文-新加坡（简体）；

——zh-tw　中文-台湾（繁体）。

如果上述代码不能满足需要，可以经权威机构注册后加以扩充，如用“zh-cn-py” 表示汉语拼音。

5.6 在内容中使用本地化属性

为了便于本地化，需要规定一些用于本地化的属性，例如：

——translate：指明哪些内容需要翻译；

——localize：指明是否需要做本地化处理；

——locID：为本地化应用建立元素的语义关联；

——locNote：通过它给翻译人员提供为理解原文应查看的重要信息。

例如，元素的内容可能既有可翻译的部分，也有不可翻译的部分，采用本地化属性可指明如下：

```
〈p〉
  〈span translate="yes"〉The XML specification is maintained by the 〈/span〉
  〈span translate="no"〉World Wide Web〈/span〉
〈/p〉
```

指示“World Wide Web”不被翻译。

此为试读，需要完整PDF请访问：www.ertongbook.com

另外，在非 XML 文件中，也可使用类似的本地化机制，例如，在 JavaScript 中插入：

```
/ * loc:span localize='no' * / … / * /loc:span * /
```

5.7 属性值中避免使用可翻译的文本

属性值的翻译会带来一些问题。首先，指定属性的本地化信息十分困难；其次，属性没有显式的 ID 供等价的属性建立关联；再者，属性值中与语言相关的空白不易处理；另外，元素 xml:lang 的作用域不覆盖属性值。

5.8 避免条件翻译

在设计 schema 时，应避免使用这样的元素：其内容是否需要翻译取决于某个属性值、父元素或同级元素的内容以及其他条件。

6 组件命名原则

6.1 拼写规则

所有的名称应该使用 UCC 规则（某些情况下，属性名称可采用 LCC），而枚举值可以使用 LCC 规则。当然枚举值中的该有的大小写规则可不受此限制，例如“ChinaGNP”。

6.2 缩写规则

可读性比起标记的长度更为重要，在元素、属性和类型名称中应尽量少用缩略语和首字母缩略语，除非这些缩略语已经为世人公认，或者在注册库中有明确的定义。例如：不要使用 DptID 作为元素或属性的名称，而应该使用 DepartmentID。如果使用缩略语，应该通过注释给出其非缩写的形式。当然也要避免使用太长的名称。

6.3 通用化名称使用的限制

在为全局（Global）元素或属性、枚举值命名时，除非它们确实是较高抽象级别上的概念，不要使用过于通用化的名称（例如：“Name”、“Address”等），应该把名称加上特定的上下文，使其更符合当前的含义（如“ReceiverName”、“EmailAddress”）。

6.4 用 URI 作为名称型数值的定义

用 URI 来唯一标识某个名称，可给它赋予更准确的内涵（这种内涵可以在专门的文档中详细描述）。例如：用“http://www.books.org/owner”来为一个“Owner”角色命名。

6.5 避免发明新名称

在为元素、属性、枚举值命名时，必须使用标准或其他规范中定义的标准名称，或者是其他外部标准文档中定义的词汇，尽量少创造新的名称。在 schema 中使用到的外部词汇也必须在注册库中注册。

如果必须创造新的名称（元素、属性、类型、枚举值等的名称），这些名称必须在行业领域专家的指导下定义，应该经过各方用户同意，并在注册库中注册。

对于 DTD 中的实体引用的命名，也应避免随意使用新的名称，应该首先考虑使用 GB/T 14814 等标准给出的实体名称。

6.6 尽量重用已有组件

schema 的制定者应尽量使用在注册库中已经定义好的标准组件（例如：元素、属性、类型等），避免制定新的雷同标准。因此，在制定 schema 之前，应该先检索注册库，看是否有可用组件存在。如果使用了注册库之外的其他标准化组织的 XML schema，也应该把它们注册到注册库中。

可使用〈include〉或〈import〉来重用已有的组件。也可使用 schema 的继承机制通过扩展已有的元素类型来定义新的元素，以提高设计效率。

为了方便重用，每个单一的 schema 不宜太大。

在 schema 设计中，如果多次用到相同的类型、元素结构、内容模型片断、属性组合等，应该将它们定义成命名的全局类型、元素、模型组和属性组，并使用通用的名称来命名。类型的定义应考虑如何方便后续的重用。

6.7 XML 组件的命名

XML 组件的命名应该遵循 GB/T 18391 的规定：XML 元素和属性名称以及 XML 数据类型名称都应遵从 GB/T 18391 相应的命名方法。另外，GB/T 18391 的名称需要经过适当转换才能用于 XML 组件命名，即除了要去除空格和分割点之外，还需将名称转换成 UCC 的形式。

简单数据类型的名称应以“……类型”或“…Type”结尾，复杂数据类型的名称应以“……结构”或“…Structure”结尾，以区别简单类型、复杂类型和元素。

6.8 元素和属性的名称应不依赖显现

元素的名称应该反映逻辑语义，而不是其显现形式。合适的名称将有助于正确理解文档并对其作适当处理。例如：

```
〈Paragraph〉〈Light〉This text is 〈/Light〉〈Bold〉important〈/Bold〉〈/Paragraph〉
```

这种表示方法就不如：

```
〈Paragraph〉〈Regular〉This text is 〈/Regular〉〈Emphasis〉important 〈Emphasis〉〈/Paragraph〉
```

因为翻译成本地化语言（比如中文）后，重点强调的内容可能不是用粗体来表示，而是加着重号来表示。显然第一种表示方法就不适合这么做。

7 命名空间使用原则

7.1 所有的 XML 文档都应使用命名空间

命名空间是在一个 XML 文档中使用多词汇表的标准的方法。即使没有命名冲突，也能使程序快速无误地确定哪个元素属于哪个应用。新开发的 XML 文档都应该为元素定义命名空间，即使词汇表的作者将来并不打算将这个词汇表用于其他应用，但不能保证词汇表的用户不会这么做。

7.2 使用统一的命名空间命名规则

应制定统一的、结构清晰的命名空间命名规则。命名空间应该用绝对路径表示的 URI 形式来指定，例如：“http://www.w3.org/TR/2001/REC-xmlSchema-2-20010502”，不建议命名空间 URI 使用收尾的斜线（“/”）。不建议使用 URN 作为命名空间的名称。应规定统一的 Schema 存放位置（schemaLocation），并将其注册到注册库中。

命名空间字符串的比较应该按大小写相关的方式逐个字符比较两个 URI 字符串（不处理 16 进制转义序列）。如果完全相同，才认为两个命名空间是相同的。

尽管命名空间 URI 不一定要对应一个真实的资源（网页、主机、域名），但如有可能，应该在那里放置一个 RDDL 文档，使命名空间的用户能获得关于该命名空间的必要信息。

7.3 使用一致的命名空间前缀

对来自外部的命名空间，使用规定的前缀，例如：

——xml　　（在 XML 标准中定义）；

——xmlns　　（在 XML Namespaces 标准中定义）；

——xsd/xs　　http://www.w3.org/2001/XMLSchema ；

——xsi　　http://www.w3.org/2001/XMLSchema-instance；

——xlink/xl　　http://www.w3.org/1999/xlink；

——xsl　　http://www.w3.org/1999/transform。

7.4 ［适用于 Schema］使用缺省的命名空间

在 Schema 文档中，如果存在目标(target)命名空间，应该将缺省(default)命名空间指定为目标命名空间。不要使用 W3C Schema 的命名空间作为缺省命名空间。

7.5 ［适用于 Schema］不要使用不带命名空间属性的导入(import)

所有导入(import)的内容必须带有相应的命名空间，以方便调试和阅读。尽可能避免使用空的、无属性的〈import〉，这会导致 Schema 难以调试和更新。

7.6 ［适用于 Schema］隐藏或暴露命名空间的准则

通过将 elementFormDefault 设置为 unqualified 或 qualified，可以实现对命名空间的隐藏或暴露。但由于全局元素是不能隐藏命名空间的，因此，如果想通过 elementFormDefault 作为暴露或隐藏命名空间的“开关”，则应尽量少用全局元素。

当实例文档的简洁性和可读性是第一位的时候，最好隐藏命名空间。因为元素附加的命名空间信息会给不熟悉 XML 的用户带来困惑。另外，当需要 Schema 有一定的灵活性，即要求对 Schema 的某些改动不影响实例文档的情况下，也可以通过隐藏命名空间来做到。例如，将原先外部命名空间定义的元素改为在 schema 内部定义(inline 方式)，这时，采用隐藏命名空间的方式就不需要改动实例文档。

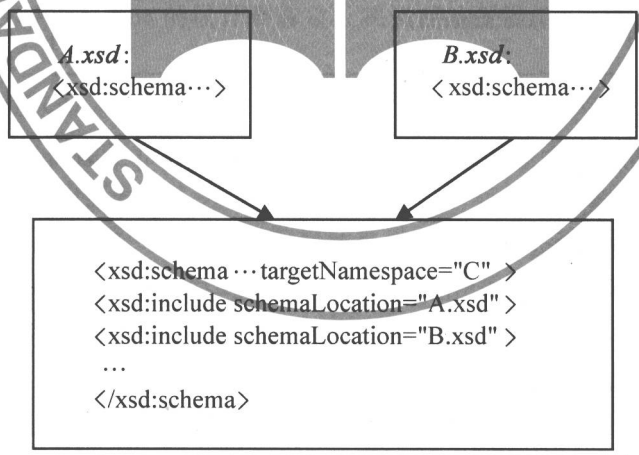
当实例文档需侧重考虑元素的拥有/承袭关系的时候(如版权的需要)，最好使用暴露的命名空间。另外，当元素同名不同义的时候，需要暴露命名空间以消除歧义。再有，当元素的处理需要命名空间的知识作辅助的时候，也应暴露命名空间。

在应用中，通常将 elementFormDefault 设置为 qualified，将 attributeFormDefault 设置为 unqualified。根据命名空间规范，没有命名空间限定的属性并不是缺省地属于其元素的命名空间，所以需要时应该显式地(explicitly)在属性前使用命名空间。

7.7 正确设计命名空间

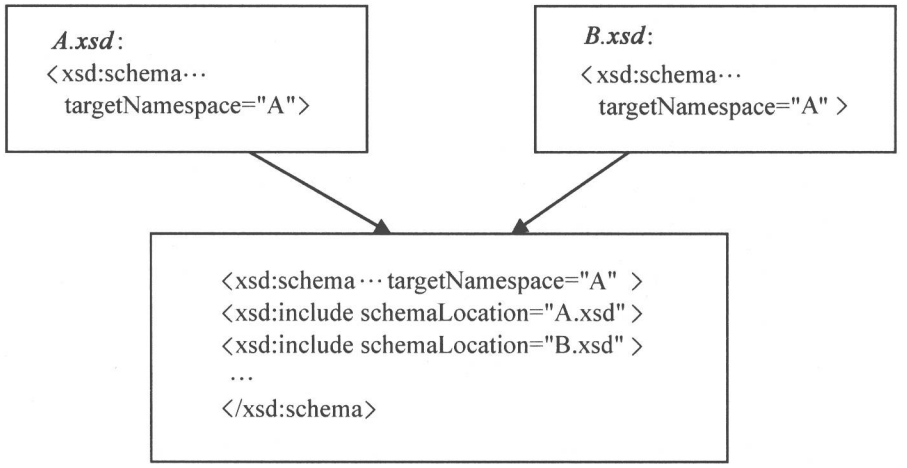
命名空间的设计大致有三种方法：

方法 1：多个 schema 中有一个“主”schema，其他的为辅助 schema。只指定主 schema 的 targetNamespace。例如：



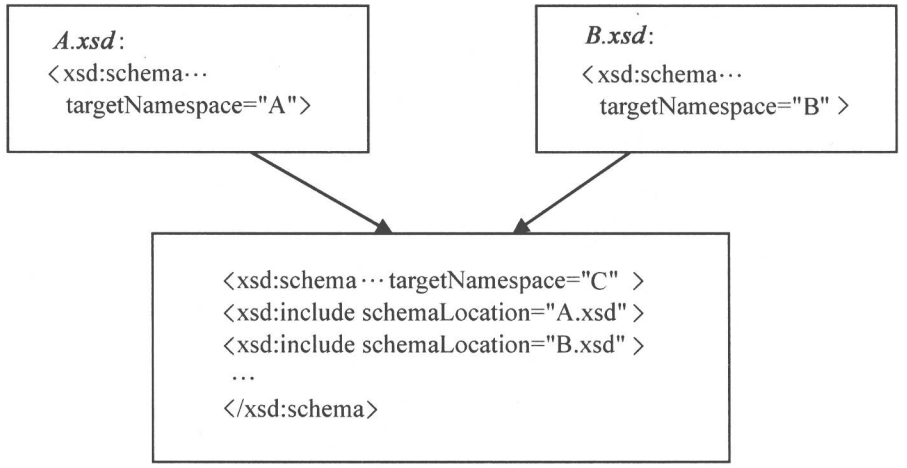
如果某些 schema 仅包含类型定义而不含元素定义，则那些 schema 最好定义成上述不带 targetNamespace 的辅助 schema。

方法 2：所有 schema 用一个 targetNamespace。如：



这种设计适合下列情况：所有的 schema 在逻辑上是紧密相关的；实例文档中没有必要显式区分元素和属性的来源及其关联，亦即不需要对元素或属性划分类别。

方法 3：所有的 schema 都有不同的 targetNamespace。如：



这种设计适合下列情况：当有多个元素具有相同的名称，为防止名称冲突；实例文档中有必要显式区分元素和属性的来源及其关联。

7.8 ［适用于 Schema］未定因素尽量靠后处理

不要把所设计的 Schema 硬性作为“targetNamespace”，一开始可考虑设计没有“targetNamespace”的 Schema，而让 Schema 的用户来决定对于具体应用应该用什么“targetNamespace”，应将 Schema 和 targetNamespace 的绑定尽量延后。

如果要导入(import)带有命名空间的 Schema，先不要硬性指定它。例如，假定有一个声明的元素使用了另一个命名空间的类型，即：〈xsd：element name＝“MyElement” type＝“n：MyElement-Type”/〉，其中 MyElementType 来自于需要〈import〉的另一个命名空间。而一旦指定了〈import〉的 schemaLocation 属性，意味着定义 MyElementType 的命名空间被严格确定了。但是，实际上没必要这样做。如果在 Schema 中不指定 schemaLocation，而让实例文档的作者来指定 MyElementType 所在的 Schema，会提高 Schema 的动态适应能力。因此，可以把类型引用和类型定义之间的绑定留到最后时

刻——在验证 Schema 之前来确定。

8 词汇表编写原则

8.1 DTD 到 Schema 的过渡

文件类型定义(DTD)用来描述一个特定目的的置标语言(词汇表),亦即定义一类文档的整体结构以及语法。DTD 源于 SGML,是 XML1.0 标准的重要组成部分。DTD 有很强的形式化特点,结构精简,但由于 DTD 采用非 XML 的语法规则,数据类型不丰富,扩展性较差等等,目前,越来越多的词汇表采用 Schema 来描述(包括 W3C 以外的 Schema)。Schema 本身是 XML 的一种应用,它使用 XML 语言来定义 DTD 的内容,充分发挥出 XML 自描述性的优势。与 DTD 相比,W3C Schema 具有一致性、扩展性、易用性、规范性和可互换性等优点。因此,文档词汇表的编写应该尽量使用 W3C Schema。但是,由于 DTD 是 XML1.0 标准所支持的,历史较早、存在大量的应用,今后在相当长的时间内,应该允许 DTD 和 Schema 并存,XML 的处理工具应该能够同时支持这两者。而在新建词汇表的时候,以及将已有 DTD 更新的时候,建议逐渐过渡到 Schema 的形式。

8.2 尽量使用 W3C Schema

除了 W3C 制定的 Schema,还存在多种 schema,例如:DTD、RELAX NG、Schematron 等。它们各自都有优缺点。但是,为了方便信息共享,最好仅使用一种 schema。一般情况下,XML 应用应该尽量使用 W3C Schema 和 DTD。

8.3 尽量使用简单的 schema 语法

XML DTD 和 Schema 都提供了强大而灵活的功能。要描述一类文档,schema 可能很简单,也可能很复杂。而大多数 schema 的用户可能都不是 XML 方面的专家,因此应尽量使用最简单的语法来描述文档,尽量不采用 schema 中不常用的机制,这对文档验证也有好处。因为有些验证工具不支持不常用的东西。

8.4 XML 文档的设计要反映信息的结构

置标语言的初衷是将文档的内容和表现形式分离。文档的内容最为人们所关心,需要能够定义文档的逻辑结构(信息模型),验证一个具体的文档正确与否,能够检索特定的文档组成部分,或者将文档的部分或全部进行传送。这些都是 XML 所擅长处理的。而与 XML 文档的结构相比,文档表现形式或文档用途是易变的。DTD 和 Schema 的设计要着重反映文档的逻辑结构,而不是文档的显现与处理。

8.5 文档的 XML 粒度

一个文档的 XML 粒度(即 XML 置标文档内容彻底与否)取决于该类文档信息建模的粒度。例如,对于一个通过三维图形交互的应用,XML 应该能够描述该图形中的各个对象,即需要 schema 能够描述图形元素,因而 XML 文档中需要加入类似 SVG 的代码。而对于产品图录这样的应用来说,一般不需要对图形进行刻画和理解,因而可以将图形数据转换成二进制数据插入到 XML 文档之中。对于数据建模不关心的部分尽可以使用 CDATA 机制或通过链接外部资源来处理。类似的情况还包括表格、代码和各种多媒体对象。一般来说,将一个文档的结构分解过细的结果要好于分解过粗,因为对于经过置标的文档结构,如果不需要处理细节,可以直接处理父元素或祖先元素节点,或合并子元素,不会有什么问题。但是对于分解不彻底的置标文档结构,如果需要处理细节就会很困难。

8.6 尽量由词汇表来规定 XML 文档的合法性

元素的各种关系、数据类型、取值范围等应该尽量使用 schema 来规定。有些内容 schema 确实无法表达时,才求助于外部应用程序。

8.7 采用 XSLT 和其他 schema 来表达事务处理规则

W3C Schema 并不适合表达复杂的事务处理规则,可以考虑通过以下三种途径来解决:

——采用其他的 schema，如 Schematron；

——编写脚本代码表达附加的限定；

——将限定规则表达为 XSLT/XPath 式样单。这种情况下，采用 schema 检验尽量多的限定条件，而采用式样单检验其余的限定条件。如果 schema 的有效性验证工具和 XSL 处理器都产生正确的结果，则实例文档是有效的。

8.8 尽量提高词汇表的可读性

schema 不仅用来验证 XML 文档，更是一种严谨的信息交流机制，系统开发人员往往需要通过阅读 schema 来全面理解和使用 XML 文档，因此必须尽可能地提高 schema 的可读性。例如：

——在名称中尽量少用缩略语。

——枚举数值也应该使用名称，而不是数值。元素和属性的命名规则同样适用于枚举值。

——避免使用元素或属性的缺省值。

——使正则表达式具有更好的可读性，例如：使用“[0-9]”，而不是“\d”。

——避免使用 xsd:redefine，以避免重用组件的副作用，提高透明性和可读性。

8.9 词汇表应尽量做到与应用程序无关

schema 的任何地方都不应存放与特定处理程序相关的配置信息，例如：SQL 语句片断、调用函数名称等。这样做会妨碍 XML 在异构系统中的使用。schema 的设计应该尽量使之能用于多种应用场景。

8.10 不宜使用处理指令来表达数据信息

尽管 XML 处理指令(Processing Instructions)提供了一种控制程序的捷径，但它毕竟只能针对特定的程序，不利于在程序间交换。而且它不能使用标准的 XML 语汇来表达信息，即处理指令的内容是不能由 Schema 来规定的。处理指令应该仅限于局部的个别的处理，处理指令不应与文档内容联系紧密，作为单一指令也不应有结构性的扩展。在考虑使用处理指令之前，应首先考虑使用元素置标。

8.11 使 XML 文档具有可扩展性

在很多情况下，标准化的 XML 文档的更新速度远远慢于现实的变化。因此，经常会发生在 XML 文档中写入了新元素，而这些元素还没有被相应标准正式承认的情况。schema 的设计要容纳开放的内容，即在实例文档中允许包含 schema 没有声明的内容。可以使用 any 和 anyAttribute 来容纳未知的元素和属性，提高 schema 的可扩展性。

在 Schema 中使用〈any〉时，要注意避免在实例文档中出现非确定性内容模型。例如，假设有以下 schema：

```
〈xsd:complexType name="Book"〉
  〈xsd:sequence〉
    〈xsd:any namespace="##any" minOccurs="0" maxOccurs="2"/〉
    〈xsd:element name="Title" type="xsd:string"/〉
    〈xsd:element name="Author" type="xsd:string"/〉
    〈xsd:element name="Date" type="xsd:string"/
    〈xsd:element name="ISBN" type="xsd:string"/〉
    〈xsd:element name="Publisher" type="xsd:string"/〉
  〈/xsd:sequence〉
〈/xsd:complexType〉
```


此为试读，需要完整PDF请访问：www.ertongbook.com