科海图书
25年·IT技术出版专家

“完全开发指南”系列编程图书
国内资深Web开发人员和培训讲师倾情奉献

开发指南

Web开发者的必读之作

指南

孙更新 裴红义 杨金龙 编著

XML 完全开发指南

为准备学习XML和XML应用程序开发者提供“一站式学习方案”

- XML基础、DTD、XSD、XPath语法
- XML文档显示——CSS、XSL
- 可扩展链接技术——XLink、XPointer
- SQL Server 2000中的XML操作
- XML DOM编程模型
- Java中的XML编程
- JavaScript中的XML编程
- .NET中的XML编程
- .NET中的XML Web Service
- XML最新应用——Ajax

光盘内容
书中所有范例源代码

科学出版社

北京科海电子出版社
www.khp.com.cn

XML 完全开发指南

孙更新　裴红义　杨金龙　编著

科学出版社
北京科海电子出版社

内 容 提 要

XML 是新一代网络数据表示、传输和交换的标准，是 Internet 环境中跨平台的、依赖于内容的技术。它的应用已经渗透到与网络数据处理相关的各个领域。

本书系统阐述了进行 XML 开发所涉及的相关技术，力图向读者展示一个完整的 XML 开发环境。主要内容包括：XML 开发工具，XML 语法，DTD 和 XSD，CSS 和 XSL，在 SQL Server 2000 中如何操作 XML 数据，XLink 和 XPointer，DOM 模型，Java、JavaScript、.NET 中的 XML 编程，XML Web Service 技术，全书最后介绍了 Web 2.0 时代 XML 技术的最新应用——Ajax。

本书采用理论与实践结合、相互渗透、逐步引导的讲解方法。在介绍技术的基础知识后，通过实例深入剖析技术的具体应用，帮助读者快速入门并逐步精通。

本书面向 XML 初学者，可作为高等院校计算机、电子商务以及信息类相关专业课程的教材，也可供广大 Web 应用程序开发者和用户参考。

图书在版编目（CIP）数据

XML 完全开发指南/孙更新，裴红义，杨金龙编著.
—北京：科学出版社，2008
ISBN 978-7-03-021174-3

Ⅰ.X… Ⅱ.①孙… ②裴… ③杨… Ⅲ.可扩充语言，XML—程序设计—指南 Ⅳ.TP312-62

中国版本图书馆 CIP 数据核字（2008）第 023775 号

责任编辑：何立兵 / 责任校对：李玉茹
责任印刷：科 海 / 封面设计：林 陶

科学出版社出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

北京艺辉印刷有限公司印刷

科学出版社发行 各地新华书店经销

2008 年 5 月第一版 开本：16 开
2008 年 5 月第一次印刷 印张：36.25
印数：000 1-4 000 字数：882 千字

定价：63.00 元（含 1CD 价格）

前　言

XML 是 Extensible Markup Language 的简写，是一种扩展性标记语言。在 Web 编程、新型数据库系统、计算机网络编程、网络数据交换和跨平台编程中，XML 正发挥着越来越重要的作用。它必将成为未来电子商务和数据管理的核心技术。

由于 XML 的重要性日益增强，XML 相关技术越来越多、越来越完善，它们共同构成了 XML 的开发环境。相关技术人员对 XML 知识的需求也在不断提高。为了让读者对 XML 的开发环境有一个深入的了解，以及在具体编程过程中如何应用 XML，本书较全面地介绍了 XML 及其相关技术。而且在详细介绍 XML 及其相关标准基础知识的同时，非常注重 XML 技术在编程方面的实际应用，重点突出了 XML 与各种常用编程语言的结合。

全书共分 14 章，各章主要内容如下：

第 1 章　主要介绍了 XML 的产生背景、优越性、应用方向等，XML 相关技术——字符集、样式表等，XML 开发工具。

第 2 章　主要介绍了 XML 的基础知识——XML 语法，内容包括：XML 文档结构，XML 格式的约束规则，如何声明元素和属性以及怎样使用命名空间。

第 3 章　主要介绍了 DTD，它用来描述 XML 文档的结构，确保 XML 文档是有效的。内容包括：元素、属性、实体在 DTD 中的声明语法和格式，以及在 XML 文档中使用 DTD 的方式。

第 4 章　主要介绍了 XML 模式定义的基本语法，重点介绍了 Schema 的文档结构，元素和属性在 Schema 中的声明语法以及 Schema 的使用方法。

第 5 章　主要介绍了 CSS 的基本语法，并在此基础上深入讲解如何将 CSS 应用于 XML 文档，让 XML 文档像 Web 页面那样进行显示。

第 6 章　主要介绍了可扩展样式表语言中的 XSLT 语言，详细讲解如何使用 XSLT 来显示 XML 文档。可扩展样式表语言主要包括 XSLT、XPath 和 XSLF 三种语言。

第 7 章　主要介绍了 Microsoft SQL Server 2000 及其插件对 XML 的支持和实现。使用 XML 技术可以方便地处理 Microsoft SQL Server 2000 中的数据。

第 8 章　主要介绍了 XLink 和 XPointer 的相关概念和用法。

第 9 章　主要介绍了文档对象模型（DOM）的结构以及在程序中如何使用这种技术操作 XML 文档。

第 10 章　主要介绍了 Java 中解析和操作 XML 文档的三种不同方式，以及每种方式下具体的编程方法。

第 11 章　主要介绍了在 JavaScript 脚本程序中利用 DOM 提供的对象和方法来操作和验证 XML 文档的具体步骤和编程模式。

第 12 章　主要介绍了在.NET 平台上操作和处理 XML 文档要用到的具体技术及相关对象，重点介绍了.NET 平台上的 DOM 编程方法，以及 XML 与 ADO.NET 技术的关系。

第 13 章　主要介绍了什么是 XML Web Service，如何创建及使用 XML Web Service 等内容。

第 14 章　主要介绍了 XML 的最新应用——Ajax。内容包括：XML 在 Ajax 中的使用，如何开发及使用“Ajax 应用”等。

本书技术阐述与实践应用相结合，强调理论联系实际开发需求，并始终以介绍 XML 中已成熟的标准和应用技术为主要出发点。书中的应用实例均取自实际开发项目，读者可以对其稍加修改后直接应用到自己的开发中。

本书既可以作为高等院校计算机、电子商务以及信息类相关专业课程的教材，也可以作为广大 Web 应用程序开发者和用户的参考资料。

本书由孙更新、裴红义、杨金龙编著，此外，参加本书编写工作的还有宾晟、王寿苹、聂江武、满在龙、杜娜、王子斌、周亦辛、仇锫铭、孙海伦、计晓斐、杨永、王萍萍、孙强，在此一并表示感谢。

由于 XML 相关技术标准的不断发布和更新，加之时间仓促，书中内容难免有纰漏和不足之处，恳请各位同仁和读者批评指正。

编　者

2008 年 4 月

目　录

第 1 章　XML 概述 1

1.1　什么是 XML 1
1.1.1　XML 是元标记语言 2
1.1.2　XML 描述的是结构和语义，而不是格式 3
1.2　XML 的产生背景 4
1.2.1　电子数据交换简介 4
1.2.2　XML 的产生及其与 SGML、HTML 的关系 5
1.3　XML 的优越性 7
1.4　XML 应用综述 8
1.5　XML 软件 9
1.5.1　XML 浏览器 9
1.5.2　XML 编辑器 10
1.5.3　XML 解析器 10
1.6　XML 相关技术 11
1.6.1　级联样式表 11
1.6.2　可扩展样式表语言 12
1.6.3　URL 和 URI 12
1.6.4　XLink 和 XPointer 12
1.6.5　Unicode 字符集 13
1.6.6　如何将这些技术融合在一起 13
1.7　XML 开发工具 13
1.7.1　XMLSpy 2007 的主要功能 14
1.7.2　XMLSpy 2007 的图形用户界面 16
1.7.3　XMLSpy 2007 的安装 19
1.7.4　XMLSpy 2007 的使用 21
1.8　本章小结 22

第 2 章　XML 语法 23

2.1　XML 文档结构 23
2.2　XML 文档规则 24
2.2.1　格式良好的 XML 文档规则 24
2.2.2　格式良好的 XML 文档 28
2.2.3　有效的 XML 文档 28
2.3　XML 声明 29
2.4　文档内容 32
2.4.1　XML 元素 32
2.4.2　XML 属性 36
2.4.3　注释 38
2.4.4　字符引用和实体引用 39

2.5 命名空间 41
2.5.1 命名冲突 41
2.5.2 解决命名冲突的方法 42
2.5.3 使用命名空间 42
2.6 XML 文档高级应用 44
2.6.1 XML 专用标记——处理指令 44
2.6.2 XML 专用标记——CDATA 节 44
2.7 本章小结 47
第 3 章 XML 文档类型定义——DTD 48
3.1 什么是 DTD 48
3.2 为什么要使用 DTD 49
3.3 DTD 声明 49
3.3.1 内部 DTD 50
3.3.2 外部 DTD 50
3.4 DTD 语法 52
3.4.1 元素声明 52
3.4.2 属性声明 60
3.4.3 实体声明 68
3.5 本章小结 70
第 4 章 XML 模式定义——XSD 71
4.1 Schema 简介 71
4.2 为什么要使用 Schema 72
4.3 Schema 的文档结构 73
4.4 XSD 的数据类型 78
4.4.1 简单数据类型 78
4.4.2 复杂数据类型 83
4.5 Schema 中的元素声明 83
4.5.1 简单元素的声明 83
4.5.2 复杂元素的声明 84
4.5.3 匿名类型定义 89
4.6 Schema 中的属性声明 91
4.7 全局元素和全局属性 96
4.8 在 XML 模式中创建元素和属性组 98
4.8.1 sequence 元素 98
4.8.2 choice 元素 100
4.8.3 group 元素 102
4.8.4 all 元素 104
4.8.5 attributeGroup 元素 107
4.9 在一个 XML 模式中使用另一个模式 109
4.9.1 include 元素 109
4.9.2 import 元素 113
4.10 本章小结 117

第 5 章　XML 文档的显示——CSS 118

5.1　样式表简介 118
5.2　级联样式表 CSS 119
5.2.1　什么是 CSS 119
5.2.2　样式表与文档的链接 120
5.2.3　级联过程 121
5.3　CSS 语法 122
5.4　CSS 属性 127
5.4.1　字体属性 127
5.4.2　文本属性 129
5.4.3　背景属性 131
5.4.4　定位属性 132
5.4.5　尺寸属性 133
5.4.6　布局属性 133
5.4.7　外补丁属性 135
5.4.8　轮廓属性 135
5.4.9　边框属性 136
5.4.10　内容属性 137
5.4.11　内补丁属性 138
5.4.12　列表属性 138
5.4.13　表格属性 139
5.4.14　其他属性 140
5.5　本章小结 142

第 6 章　XML 文档的显示——XSL 143

6.1　XSL 概述 144
6.2　XSLT 145
6.2.1　为什么要用 XSLT 145
6.2.2　XSLT 的历史 146
6.2.3　XSLT 和 CSS 的比较 146
6.3　XPath 146
6.3.1　XPath 定义 147
6.3.2　XPath 数据类型 147
6.3.3　XPath 表达式 149
6.3.4　XPath 定位路径 152
6.3.5　XPath 标准函数库 157
6.4　XSL 文档结构 160
6.5　XSLT 的元素语法 164
6.5.1　XSL 模板 164
6.5.2　使用 xsl:value-of 获得节点值 169
6.5.3　使用 xsl:for-each 处理多个元素 176
6.5.4　默认的模板规则 181
6.5.5　对输出元素排序 182

6.5.6 选择 184
6.5.7 XPath 表达式在 XSL 样式表中的使用方法总结 194
6.5.8 决定输出要包含的内容 199
6.5.9 使用 xsl:copy 复制当前节点 207
6.5.10 使用 xsl:number 为节点计数 211
6.5.11 使用 xsl:variable 定义常数 216
6.5.12 命名模板 218
6.5.13 参数及使用 219
6.5.14 删除和保留空白 224
6.5.15 合并多个样式表 224
6.6 本章小结 230
第 7 章 XML 和 Microsoft SQL Server 2000 的集成 231
7.1 Microsoft SQL Server 2000 对 XML 的支持 231
7.1.1 SQL Server 2000 中的 XML 特征 231
7.1.2 SQL Server 的 XML 体系结构 233
7.1.3 启用 IIS 对 SQL Server XML 的支持 233
7.2 使用存储在 SQL Server 中的数据生成 XML 文档 236
7.2.1 For XML Raw 模式 237
7.2.2 For XML Auto 模式 238
7.2.3 For XML Explicit 模式 238
7.2.4 转义字符和特殊符号 239
7.3 对 SQL Server 数据库数据执行 XPath 查询 239
7.3.1 XDR 架构元素 240
7.3.2 XPath 查询的步骤 244
7.4 把 XML 数据插入到 SQL Server 数据库表中 246
7.5 本章小结 250
第 8 章 XLink 和 XPointer 251
8.1 XLink 251
8.1.1 链接 251
8.1.2 简单链接 253
8.1.3 扩展链接 254
8.1.4 外联链接 256
8.1.5 扩展链接组 256
8.2 XPointer 259
8.2.1 绝对位置项 260
8.2.2 相对位置项 264
8.2.3 字符串位置项 266
8.2.4 origin 绝对位置项 267
8.3 本章小结 268
第 9 章 XML DOM 编程模型 269
9.1 文档对象模型概述 269

9.2　XML 解析器 270
9.3　DOM 解析树 271
9.4　MSXML DOM 模型结构 273
9.4.1　DOMDocument 对象 274
9.4.2　IXMLDOMNode 对象 276
9.4.3　IXMLDOMNodeList 对象 277
9.4.4　IXMLDOMParseError 对象 277
9.5　DOM 编程步骤 277
9.6　本章小结 279
第 10 章　Java 中的 XML 编程 280
10.1　使用 DOM 解析 XML 280
10.1.1　Java DOM 的 API 281
10.1.2　Java DOM 的应用 283
10.2　使用 SAX 解析 XML 289
10.2.1　SAX 中的事件 289
10.2.2　Java SAX 的 API 292
10.2.3　Java SAX 的应用 293
10.3　使用 JDOM 解析 XML 297
10.3.1　JDOM 的 API 297
10.3.2　JDOM 的应用 299
10.4　本章小结 303
第 11 章　使用 JavaScript 操作 XML 文档 304
11.1　JavaScript 语言概述 304
11.1.1　JavaScript 的基本语法 305
11.1.2　JavaScript 事件 306
11.1.3　JavaScript 程序的编写及运行 308
11.2　使用 JavaScript 解析 XML 文档 309
11.2.1　创建 DOM 文档对象并载入 XML 文档 309
11.2.2　遍历 XML DOM 文档 311
11.2.3　在 DOM 中添加、删除和替换节点 315
11.3　使用 JavaScript 验证 XML 文档 318
11.3.1　使用 DOM 验证 XML 文档对于 DTD 的有效性 319
11.3.2　使用 DOM 验证 XML 文档对于 XML 模式的有效性 324
11.4　使用样式表实时处理 XML 文档 330
11.5　本章小结 340
第 12 章　.NET 中的 XML 编程 341
12.1　使用流模式处理 XML 文档 342
12.1.1　读取 XML 文档 342
12.1.2　写 XML 文档 365
12.2　使用 DOM 处理 XML 文档 371
12.2.1　.NET W3C DOM 类简介 371

12.2.2 使用 DOM 加载及保存 XML 数据 373
12.2.3 使用 DOM 浏览 XML 文档 374
12.2.4 创建新节点 397
12.2.5 修改和删除节点 408
12.3 ADO.NET 与 XML 418
12.3.1 ADO.NET 简介 418
12.3.2 XML 与 DataSet 对象的关系 419
12.3.3 使用 DataSet 对象访问 XML 文档 420
12.3.4 同步 XML 文档和 DataSet 对象 435
12.4 XML 查询 437
12.4.1 XPathDocument 对象 438
12.4.2 XPathNavigator 对象 438
12.4.3 XPathExpression 对象 441
12.5 XML 转换 446
12.5.1 使用编程的方式实现 XML 转换 446
12.5.2 使用 ASP.NET 中的控件显示 XML 数据 449
12.5.3 扩展 XSLT 样式表 452
12.5.4 扩展对象 475
12.6 XML 序列化 480
12.6.1 XML 的序列化 481
12.6.2 使用 XmlSerializer 序列化对象 490
12.6.3 控制 XML 序列化 497
12.7 本章小结 506

第 13 章 .NET 中的 XML Web Service 507

13.1 XML Web Service 概述 507
13.1.1 什么是 Web Service 507
13.1.2 Web Service 软件的支持 508
13.1.3 XML Web Service 的定义 508
13.1.4 XML Web Service 的技术支持 510
13.2 创建和使用 XML Web Service 513
13.2.1 创建 XML Web Service 513
13.2.2 使用 XML Web Service 520
13.3 本章小结 533

第 14 章 XML 最新应用——Ajax 534

14.1 Ajax 概述 534
14.2 Ajax 技术基础 535
14.3 Ajax 技术核心 536
14.3.1 创建 XMLHttpRequest 对象 536
14.3.2 XMLHttpRequest 对象的方法与属性 537
14.3.3 Prototype 框架 539
14.4 Ajax 应用 552
14.5 本章小结 570

1 XML 概述

章前导读

XML 使用一个简单而又灵活的标准格式，为基于 Web 的应用提供了一个描述数据和交换数据的有效手段。本章将介绍什么是 XML，XML 产生的背景，XML 的优越性和 XML 的应用，以及 XML 软件和相关的技术及开发工具。

学习重点

- ✪ 什么是 XML
- ✪ XML 的优越性
- ✪ XML 的应用
- ✪ XML 开发工具

1.1 什么是 XML

XML（Extensible Markup Language，可扩展的标记语言）是一套定义语义标记的规则，这些标记将文档分成许多部件并对这些部件加以标识。它也是元标记语言，可以定义其他与特定领域有关的、语义的、结构化的标记。

XML 是 Internet 环境中跨平台的、依赖于内容的技术，是当前处理分布式结构信息的选择工具。它可以简化通过 Internet 的文档信息传输。

XML 由 XML 工作组（原先的 SGML 编辑审查委员会）开发，此工作组由 World Wide Web Consortium（W3C）在 1996 年主持成立。工作组由 Sun Microsystems 的 Jon Bosak 负责，同样由 W3C 组织的 XML SIG（Special Interest Group，原先的 SGML 工作组）积极参与了 XML 工作组的工作。

XML 的设计目标如下：

（1）XML 应该可以直接用于 Internet。
（2）XML 应该支持大量不同的应用。
（3）XML 应该与 SGML 兼容。
（4）处理 XML 文件的程序应该容易编写。
（5）XML 中的可选项应无条件地保持最少，理想状况下应该为 0 个。
（6）XML 文件应该是人可以直接阅读的，应该是条理清楚的。

（7）XML 的设计应快速完成。
（8）XML 的设计应该是形式化的、简洁的。
（9）XML 文件应易于创建。
（10）XML 标记的简洁性是最后考虑的目标。

这些目标中有几点值得注意。

第一，W3C 希望 XML 是非常简单的；实际上，好几个目标都包含了“容易”和“清晰”的含义。

第二，W3C 赋予了 XML 两个目标：供人阅读和供 XML 处理程序处理。XML 处理程序或者解析器是用来处理 XML 文档的软件包。处理程序可以识别 XML 文档的内容，也可以读、写或修改已存在的文档，还可以直接创建新的文档。其目的是通过保持开发的简单性来打开 XML 处理程序的市场。严格的构造规则意味着所需要进行的处理会比较少。因而也意味着 XML 文档可应用于移动设备，诸如手机和 PDA。

通过使文档可供人阅读，程序开发者可以更方便地使用数据，也可以更容易地开发和调试应用程序。Unicode 的应用程序开发者可以使用很多不同的语言创建 XML 文档。然而，其产生的问题是 XML 文档可能变得比较啰嗦，用 XML 来描述数据也会比采用其他方式更冗长。

第三，对于 XML 文档（Document），这个术语的含义比传统意义上的物理文档更为丰富。一些 XML 文档以物理形式存在，但另一些 XML 文档是以遵循 XML 构造规则的信息流的形式创建的，例如 Web 服务以 XML 形式返回的数据库调用结果。

下面就是一段 XML 示例文档。

```
<myfile>
<title>XML Quick Start</title>
<author>ajie</author>
<email>ajie@aolhoo.com</email>
<date>20010115</date>
</myfile>
```

注意

（1）这段代码仅仅能让读者感性认识 XML，并不能实现什么具体应用。
（2）其中类似<title>、<author>的语句就是自己创建的标记（tags），它们和 HTML 标记不一样，例如这里的<title>是文章标题的意思，而在 HTML 中<title>是指页面标题。

1.1.1 XML 是元标记语言

关于 XML，要理解的第一件事是，它不是像超文本标记语言（Hypertext Markup Language，HTML）那样的语言或是格式化的程序。这些语言定义了一套固定的标记，用来描述一定数目的元素。如果标记语言中没有所需的标记，用户只能等待标记语言的下一个版本，希望在新版本中能够包括所需的标记，这样一来就得依赖软件开发商的选择。

但是 XML 是一种元标记语言。用户可以定义自己需要的标记。这些标记必须根据某些通用的原理来创建，但是在标记的意义上，具有相当的灵活性。例如，若用户正在处理与家谱有关的事情，需要描述人的出生、死亡、埋葬地、家庭、结婚、离婚等，这就必须

创建用于每项的标记。新创建的标记可在文档类型定义（Document Type Definition，DTD）中加以描述。

XML 定义了一套元句法，与特定领域有关的标记语言（如 MusicML、MathML 和 CML）都必须遵守它。如果一个应用程序可以理解这一元句法，那么它也能够理解所有的由此元语言建立起来的语言。也就是说，浏览器无需事先了解多种不同的标记语言使用的每个标记。事实上，浏览器在读入文档或它的 DTD 时才了解给定文档使用的标记。关于如何显示这些标记的内容的详细指令是由附加在文档上的另外的样式表提供的。

有了 XML 就意味着不必等待浏览器的开发商来满足用户的需要了。用户可以创建自己需要的标记，当需要时，告诉浏览器如何显示这些标记就可以了。

1.1.2 XML 描述的是结构和语义，而不是格式

关于 XML，要了解的第二件事是，XML 标记描述的是文档的结构和语义。它不描述页面元素的格式，可用样式表为文档增加格式化信息。文档本身只说明文档包括什么标记。

作为对照，HTML 文档包括了格式化、结构和语义的标记。在 HTML 中，<B>就是一种格式化标记，它使其中的内容变为粗体。<STRONG>是一种语义标记，意味着其中的内容特别重要。<TD>是一种结构标记，指明内容是表中的一个单元。事实上，HTML 中某些标记同时具有这三种意义。例如，<H1>标记可同时表示 20 磅的 Helvetica 字体的粗体、第一级标题和页面标题。

例如，在 HTML 中，一首歌可能是用定义标题、定义数据、无序的列表和列表项来描述的。但是事实上这些项目没有一件是与音乐有关的。用 HTML 定义的歌曲如下：

```
<dt>Hot Cop
<dd> by Jacques Morali Henri Belolo and Victor Willis
<ul>
<li>Producer: Jacques Morali
<li>Publisher: PolyGram Records
<li>Length: 6:20
<li>Written: 978
<li>Artist: Village People
</ul>
```

而在 XML 中，同样的数据可能标记为：

```
<SONG>
<TITLE>Hot Cop</TITLE>
<COMPOSER>Jacques Morali</COMPOSER>
<COMPOSER>Henri Belolo</COMPOSER>
<COMPOSER>Victor Willis</COMPOSER>
<PRODUCER>Jacques Morali</PRODUCER>
<PUBLISHER>PolyGram Records</PUBLISHER>
<LENGTH>6:20</LENGTH>
<YEAR> 978</YEAR>
<ARTIST>Village People</ARTIST>
</SONG>
```

在 XML 清单中没有使用通用的标记如<dt>和<li>，而是使用了具有意义的标记，如<SONG>、<TITLE>、<COMPOSER>和<YEAR>等。这种用法具有许多优点，包括源代码易于阅读，使人能够看出代码的含义。

对于这个例子，XML 标记还使程序易于找出文档中的所有歌曲。而在 HTML 中，它只能告诉人们这个元素是 dt。人们不能确定 dt 到底代表一首歌的题目还是定义，抑或只是一些设计者喜爱的缩进文本格式。事实上，dt 元素在单一文档中可能包括三种意义。

可以选择 XML 的元素名称，以便使其在附加的上下文中具有额外的意义。XML 比 HTML 更为灵活而且适用于各种应用，因为有限数目的标记不必用于许多不同的目的。

1.2 XML 的产生背景

XML 产生的意图从来就不是在软件开发中使用，至少在早期不是如此。其早期规范：XML 1.0、XPath、XSLT、Namespaces in XML、XHTML 和 DOM 并无任何内容集中关注软件开发人员的需求。XML 最初的设计目的是为了电子数据交换，更具体地说是为电子数据交换提供一个统一的标准格式。

1.2.1 电子数据交换简介

1. 什么是 EDI

EDI（Electronic Data Interchange，电子数据交换）是一种利用计算机进行商务处理的方法。EDI 将贸易、运输、保险、银行和海关等行业的信息，用一种国际公认的标准格式，通过计算机通信网络，使各有关部门、公司与企业之间进行数据交换与处理，并完成以贸易为中心的全部业务过程。

EDI 不是用户之间简单的数据交换，EDI 用户需要按照国际通用的消息格式发送信息，接收方也需要按国际统一规定的语法规则，对消息进行处理，并使其他相关系统进行 EDI 综合处理。整个过程都是自动完成，无须人工干预，减少了差错，提高了效率。因此 EDI 又被人们通俗地称为“无纸贸易”。

从上述 EDI 定义不难看出，EDI 包含了三个方面的内容，即计算机应用、通信网络和数据标准化。其中计算机应用是 EDI 的条件，通信环境是 EDI 应用的基础，标准化是 EDI 的特征。这三方面相互衔接、相互依存，构成 EDI 的基础框架。

2. EDI 的工作过程

EDI 系统由通信模块、格式转换模式、联系模块、消息生成和处理模块等 4 个基本功能模块组成。EDI 的工作过程如下：

第一步，信息的发送者使用计算机文件来聚集交易需要的数据；

第二步，组合数据输入到一个软件模块中，在其中将事务转换为 EDI 标准格式；

第三步，通过增值网络 VAN 的中间通道群，将软件模块的结果数据传送给接收者；

第四步，在接收终端，由软件模块把数据从 EDI 格式翻译成接收者的应用程序能理解的文件。

3. EDI 的好处及局限性

使用 EDI 的主要优点如下：

（1）降低了纸张文件的消费。

（2）减少了许多重复劳动，提高了工作效率。

（3）使得贸易双方能够以更迅速、有效的方式进行贸易，大大简化了订货过程和存货过程，使双方能及时地充分利用各自的人力和物力资源。

（4）可以改善贸易双方的关系，厂商可以准确地估计商品的需求量，货运代理商可以简化大量的出口文书工作，商业用户可以提高存货的效率，提高它们的竞争力。

使用 EDI 的局限性如下：

（1）交易建立在固定的交易设备上，成为发展新服务、新商业单位的瓶颈。

（2）固定的商业规则，不适应变化。

（3）高成本，中、小型企业用不起。

（4）标准发展速度缓慢，不能在不同系统间实现数据交换。

基于此，人们期待一种通用的、便于使用的新型数据交换系统。

1.2.2 XML 的产生及其与 SGML、HTML 的关系

XML 同 HTML 一样，都来自 SGML（Standard Generalized Markup Language，标准通用标记语言）。早在 Web 未发明之前，SGML 就已存在。正如它的名称所言，SGML 是国际上定义电子文件结构和内容描述的标准，是一种非常复杂的文档的结构，主要用于具有大量高度结构化数据的防卫区和其他各种工业领域，利于分类和索引。SGML 十分庞大，既不容易学，又不容易使用，在计算机上实现也十分困难。鉴于这些因素，Web 的发明者——欧洲核子物理研究中心的研究人员根据当时（1989 年）的计算机技术，开发了 HTML。

HTML 只使用 SGML 中很少的一部分标记，例如 HTML 4.0 中只定义了 70 余种标记。为了便于在计算机上实现，HTML 规定的标记是固定的，即 HTML 语法是不可扩展的。HTML 这种固定的语法使它易学易用，在计算机上开发 HTML 的浏览器也十分容易。正是由于 HTML 的简单性，使得基于 HTML 的 Web 应用得到极大的发展。

但是，随着 Web 应用的不断发展，HTML 的局限性也越来越明显地体现了出来。

（1）HTML 是一种界面技术。它把数据和数据的表现形式混在了一起，这使得分开两者变得相当的困难。

（2）它有一个复杂的标签集。人们不能用自己的特定应用标签来扩展它。

（3）它是“平面型”的。不能使用数据的层次结构来表现数据间诸如包含、重要性等关系。

（4）它不能把数据简单地传送给客户端让客户端自行进行进一步的处理。事实上，HTML 总是在服务器端产生，客户端只是一个显示机器。

（5）它只能提供一种显示方式给数据。如果想提供不同的显示方式，就不得不在服务器端重新产生这些数据。

（6）它的可读性不强，无论是人读还是计算机读都是一样。HTML 也不是很严谨，有些标签需要匹配开始符和结束符（像<body>和</body>），但是也有一些只有开始符没有结束符（像<p>和<b>）。HTML 解释器不得不处理这些有些随机的格式。

（7）链路丢失后不能自动纠正。由于许多页面的 URL 地址经常变化，当浏览这些页面时就会提示“404 URL 地址未找到”的信息。所以你不得不手工一个个地更改链接相关页面的 URL 地址，这大大加重了 Web 页面的维护工作量。

（8）动态内容需要下载的部件太多。用 HTML 建立的页面目前还不能对其页面的外观属性，例如色彩、字体、背景等实现更新，只能重新下载一个新的页面。

（9）搜索时间长。由于 HTML 页面没有类似于数据库的结构，在这样的文档资料中搜索目标时需要扫描全部页面的所有内容，往往检索出一大堆与主题词无关的内容，这是因为 HTML 无法区分信息与元信息而造成的。而且 HTML 不支持信息嵌套体系结构，因而限制了全文检索功能。

（10）HTML 缺乏对双字节或多国文字的支持，或者说支持不够。例如中文信息页面在不同的平台下会出现格式不齐等问题。

HTML 过于简单的语法严重地阻碍了用它来表现复杂的形式。尽管 HTML 推出了一个又一个新版本，已经有了脚本、表格、帧等表达功能，但始终不能满足不断增长的需求。正是在这种形势下，Web 标准化组织 W3C 建议使用一种精简的 SGML 版本——XML 应运而生了。

XML 是一个精简的 SGML，它将 SGML 的丰富功能与 HTML 的易用性结合到 Web 的应用中。

XML 取 SGML 之长。

（1）XML 是 SGML 的一个子集，继承其常用和基本运算功能，去除复杂又不常用的部分。

（2）Meta-Language（元语言）：SGML 和 XML 都可以用来制定其他标记语言。

（3）文件格式定义：继承 SGML 允许用户自定义（DTD）特性。

（4）排版样本：XML 像 SGML 一样结合排版样本（CSS）来显示网页。

XML 补 HTML 之短：

（1）HTML 控制标记是固定的，无法扩展；XML 允许用户在 DTD 中声明元素，然后在文件中即可使用。

（2）HTML 文件缺乏结构性，主要被设计成有显示数据的能力。XML 允许嵌套的信息结构，提供了直接处理 Web 数据的通用方法。

（3）使用当前的 HTML，开发者必须对文档进行许多的调整才能兼容流行的浏览器。由于浏览器无法检查 HTML 代码，因此导致 Web 上大量的文档有 HTML 语法错误。XML 对浏览器的兼容性增强，设计人员软容易实现 XML 浏览器。

但是，XML 并非是用来取代 HTML 的。HTML 着重描述如何将文件显示在浏览器中，XML 与 SGML 相近，着重描述如何将文件以结构化方式表示。就网页显示功能来说，HTML 比 XML 强，但就文件的应用范畴来说，XML 比 HTML 超出很多。