XML Developer's Handbook

XML 高级开发指南

XML

〔美〕 Kurt Cagle 著
周生炳 肖伟 译
周生炳 审校

◇ 建立XML电子商务应用程序

SYBEX
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL: http://www.phei.com.cn

XML Developer's Handbook

XML高级开发指南

〔美〕 Kurt Cagle 著

周生炳 肖 伟 译

周生炳 审 校

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 提 要

人们已日渐深入地认识到，计算机的力量不在于其计算能力而更多地在于其通信能力。因此，人们悄然把目光投向能以完全与机器无关的方式交流信息的语言——XML。

本书面向开发人员，深入探讨了微软XML分析器的各个方面，包括XML文档对象模型、XPath、XSL转换、XML模式以及XML与浏览器、ASP、微软数据库、微软电子商务工具BizTalk和程序设计等之间的关系。作者依其丰厚的XML开发背景，在书中收录了大量实用代码，提出了很多独到见解。相信一定能对读者有所帮助。

图书在版编目（CIP）数据

XML高级开发指南/（美）卡哥尔（Cagle, K.）著；周生炳，肖伟译. - 北京：电子工业出版社，2001.6
书名原文：XML Developer's Handbook
ISBN 7-5053-6760-9

Ⅰ. X… Ⅱ. ①卡… ②周… ③肖… 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（2001）第035378号

书　　名：**XML高级开发指南**
著　　者：〔美〕Kurt Cagle
译　　者：周生炳　肖　伟
审　　校：周生炳
责任编辑：李　莹
印 刷 者：北京天竺颖华印刷厂
装 订 者：三河金马印装有限公司
出版发行：电子工业出版社　URL:http://www.phei.com.cn
　　北京市海淀区万寿路173信箱　邮编：100036　电话：68279077
　　北京市海淀区翠微东里甲2号　邮编：100036　电话：68252397
经　　销：各地新华书店
开　　本：787×1092 1/16　印张：31　字数：790千字
版　　次：2001年6月第1版　2001年6月第1次印刷
书　　号：ISBN 7-5053-6760-9 / TP・3790
定　　价：52.00元

版权贸易合同登记号　图字：01-2000-3168

仅以本书献给我的两个女儿，凯蒂和珍尼弗。我今天论述的技术将是她们明天生活世界中的“日常用品”，知道我将拥有影响她们未来生活的材料使我深受鼓舞。

致　谢

这是一部大部头著作。撰写这样一部书就像攀登险峰：登上山顶的机会很小，你会发现自己淹没在尝试攀登的诸多事务中。

我对这项工作的庞杂了然于胸，因此我感谢本书的选题和组稿编辑丹尼斯·桑托罗·林肯，尤其是意识到在我次女出生后需要抽身一段时间时。同样，我感谢编辑们的努力，Sybex的技术编辑、加工编辑、校对和版面设计师帮助我使本书成书。

本书很大部分集中介绍微软MSXML 3分析器。感谢微软公司的克里斯·拉福特、查理·海克雷布斯允许我使用该分析器的早期版本，并提供大量技术支持。

曼宁出版社的杰作《利用VB和ASP进行XML程序设计》一书的作者马克和特雷塞·威尔逊一直在运作VBXML Web网站，他们既是我的好友，又是我诉苦时的耐心听众。访问他们的网站（http://www.vbxml.com），你会觉得不虚此行。

最后，感谢我的妻子安妮，她对与一个计算机著作作家结婚有点恼火，但还是保持最大的克制。

译 者 序

熟悉HTML的读者都知道，利用HTML能够方便地访问因特网资源，但它不能控制数据，而XML则擅长设计数据。它是一个格式独立、与平台和应用程序无关的语言，只要应用程序都支持XML词汇，就能在应用程序之间无缝地交换数据。因此，XML意在提供Web上的结构化信息交换机制，从而将Web从发布媒体转换为一个应用程序处理环境。

本书读者对象是开发人员。书中深入探讨了微软XML分析器的各个方面，包括XML文档对象模型、XPath、XSL转换、XML模式以及XML与浏览器、ASP、微软数据库、微软电子商务工具BizTalk和XML程序设计的关系。最后，作者展望了XML的未来，提出了很多独到的见解。本书选配光盘提供了几个XML演示软件、程序设计模块和书中所有编号程序清单，读者可借助其中的设计环境尝试运行这些程序，这将大大提高您的学习效率，加深对XML的理解。与Sybex的另一本入门读物《XML从入门到精通》（中译本已由电子工业出版社出版）配套使用本书，效果会更好。

本书由周生炳、肖伟译，周生炳审校。翻译过程中得到宋浩、王宴民、李满卫、刘晋伟、陈黎、鲍明涛、陈艳超、陈豫新、王欣、王芳、姜苗苗、吴传芝、孙岩、沈明华、黄智颖、许蕾、吕娟、扬凡、莫琳琪、李晓燕、刘笑凯等同志的协作，对他们的帮助，我们深表感谢。由于本书涉及范围广，内容新，语言不太规范，译者学识所限，错误在所难免，希望读者不吝指教。

引　言

1997年下半年，我在为出版界和私人客户开发和编写DHTML项目的背景下而首次接触XML。当时，我正在为微软公司从事一个项目，即开发一个为小企业建立Web网站的程序，该企业旨在帮助他人判断采购某些软件配置的好处。这个程序在Internet客户机上开发，使用最常用的JavaScript代码，其基本的设计理念是以一种清晰而合理的方式管理层次信息。然而，我最终却花了太多时间尝试把数据装入系统，在偶然听人提到微软（悄悄地）塞入Internet Explorer 4浏览器的一种技术的测试版（XML分析器）前，我几乎绝望了。

当然，以前我曾听人谈到过XML，当时流行的看法是它最终一定会取代HTML成为客户机语言，只要所有浏览器都装备它。当我开始使用该语言时，我发现它既粗糙又晦涩，而且要求深入了解DTD和外部实体这样的神秘概念。

为数廖廖的几本书是SGML专家撰写的，它们居高临下地谈论XML，好像它是每个人都应理解的东西，可它与真正的母语相比不过是婴儿语言。更糟糕的是，没有一本书讨论如何用该语言编程序。这也许可以理解，因为几乎没有人考虑利用XML传递数据，尽管它是表达复杂数据结构的极佳机制（正如我在本书中希望论述的那样）。

当我与开始推崇XML的SGML头面人物谈话时，他们中的大多数只是轻描淡写地说："可怜的初学者，你不能用XML编程，它只不过是一种标记语言。"能够证明他们错了一定很惬意。

第一个Microsoft XML分析器（我体验过的第一个分析器）比较粗糙，对围绕层次树一个一个节点地导航很管用，但对其他许多东西就没什么用了。虽然在某些老Internet Explorer 4浏览器中还能找到它，但该分析器在很大程度上已经废弃了（谢天谢地）。

随着微软公司的努力，现有XML分析器是功能相当强的引擎，能做出难以置信的事情，是微软最近宣布的.NET战略的动力。也许它是目前世界上最好的分析器之一。

撰写这样大篇幅的一本书很费时间，当技术仍在发展时，这种写法可能成问题。现在就是这种情况。我在本书付印之前不久撰写本引言。在少数情况下，本书其他地方所写的东西可能很快就过时了，例如，在夏天过去之前，XML Schema Definition Language可能至少要经过一轮以上的修订。

我希望本书能帮助你转变思想方法，从传统语言的过程范式转变为XML及其相关标准的声明范式。你可能发现自己在思考集合、树、路径和转换而不是特性、方法和事件。只要你稍稍熟悉它，就会对赋予你相当平淡的数据库结果的复杂SQL语句感到恼火，并发现自己对为什么不使Java和VB程序员使用同一界面感到奇怪，你会倾向于用<document>标注打开所有Word文档。

我们处在将完全重建计算机版图的一场革命的早期阶段，尤其在你认识到计算机的力量不在其"计算"能力而更多地在于其通信能力的时候。XML是以完全与机器无关的方式交流信息的一种方法。你的服务器和我的客户机可能是两家运行相互竞争操作系统的公司以迥然不同的程序设计语言编写的，但有了XML，这不是问题。

此外，XML是一种富有侵略性的技术：当你将XML引入你的系统时，可能配置它的潜在之处也变得明确无疑，即使在一个微不足道的角色中。过一会，用来编写临时INI文件的XML代码已变异成整个体系结构。作为一个混沌理论的学生，我喜欢说XML是一种浮现技术，它将演化成取代其他体系结构的体系结构。的确，你看到的是一种主要范式转变，未来几十年整个社会可能都会有反响。

在一定程度上，这是由于Internet从HTML到XML的可能转变——一种正在发生的转变。可以通过XSLT转换XML，这意味着你能够创建表达大型复杂文档逻辑结构的文档（如使用OpenDoc规范的文档），然后将它们转换成不同浏览器上的表示。在某些情况下，这些浏览器是我们最熟悉的——Internet Explorer、Netscape Navigator等等——但是，越来越多的情形是，浏览器是你手机上的一个小视窗，或你掌上设备上的一块信用卡大小的屏。利用无线应用协议（Wireless Application Protocol，WAP），这些设备已经大举进入XML世界。很快，它们还将利用语音XML语言，如VoxML。这些新形式表明把消息与媒体分开的好处，它们不过是许多接踵而来的新特征的第一批。

所以，试一试这里提供的代码。你会在程序清单标题中或其清单标题结尾处的括号中找到每个程序清单的文件名，然后在本书选配光盘的对应文件（完整的可执行文件或代码片段）找到每个程序清单。参阅Sybex的配套读物《XML从入门到精通》（已由电子工业出版社翻译出版），它更深入地考察了XML规范本身。寻找行为模式（即能帮助你理顺你的工作的思想方法），一旦你习惯了该语言，这些模式就会涌现出来。这种技术很酷——也将成为一种普遍应用的技术——它会开辟发展的所有可能性。

到哪里获取更多信息

跟上XML的发展并不总是轻而易举的，但是，一般情况下，我强烈建议你关注W3C Web网站（http://www.w3.org），尤其是该组织维护的推荐标准、草案和批复（http://www.w3.org/TR）。微软在MSDN上的XML Web网站（http://msdn.Microsoft.com/xml）提供了关于新分析器和XML技术的最新信息（我目前正在为该网站编写MSDN XSLT用户指南，所以我可以较肯定地这么说），它应是你的浏览器的一个主要书签。另外，可以访问XML.com（http://www.xml.com）了解当前行业发展情况（该网站是Edd Dumbill编辑的）。与XML相关的更详细的技术设计问题见VBXML Web网站（http://www.vbxml.com），它由Mark Wilson编辑，我的XML和XSLT文章和资源也放在该网站上。

最后，请与我联系，我的电子邮件地址是cagle@olywa.net。我不能保证我能回答你的所有问题，但我会尽力而为。

目　录

第1章　为什么使用XML

主要内容：

- 了解环境的需要
- 回顾XML的兴起
- 详述XML的作用
- 了解XML的使用

掌握可扩展标记语言（Extensible Markup Language）或XML可能改变人们对于程序设计、通信甚至一般表示的含义以及环境的理解。尽管语言本身很简单（这正是它的力量所在），但它却可能彻底改变计算机领域的版图。在本章中，你将了解为什么XML会成为明天的数据语言，以及它究竟有多重要。

对环境的需要

什么是含义？这是哲学系一年级大学生肯定要碰到的一个问题。直到最近，含义问题对计算机程序设计并无多大影响，但对该问题的回答最终将成为未来几十年计算的基石。理由很简单：我们的数据太多而其含义不明。

我们生活在一个数据无处不在的时代。从起床的那一刻开始，你就被数据所淹没——叮叮的闹钟、报道华盛顿情况的新闻广播、未来几天的天气预报。我们翻开报纸，浏览过去24小时内发生的事件，了解棒球明星的RBI（击球跑垒得分），仔细考察上次检查以来我们的共同基金又涨落了几美分。

在工作中，我们要花越来越多的时间来对付各种各样的数据，力图搞清西部地区销售数字的含义，尝试理解为什么编辑器产生错误——43225，确保列入帐单的时间分类准确（指按时间收费的律师、咨询人员等专业人员帐单中的时间分类——译注）。知识工人作为一个囊括下到数据录入员上至大公司首席执行官的一般称谓，已经取代蓝领工人，成为典型的挣工资者。计算机革命已经围绕知识、信息和含义重组整个社会。

具有讽刺意义的是，尽管很少有数据会导致对问题的错误理解，但若数据太多，就不得不投入大量精力和时间从数据中提取信息。人类历史的很长一段时期，也许直到五十年代左右，数据还是昂贵而相对稀少的，那时，人们对说不定哪天数据会太多的想法会嗤之以鼻。

过去五十年，随着创建、处理和显示数据的手段以指数速度飞速发展，这种情况完全颠倒过来了。IBM主席戈登·摩尔首先观察到的著名预言“摩尔定律”成为这种飞速发展的可靠见证：处理能力（及获取和显示数据的能力）大约每18个月翻一番。过去十年，这个速度甚至更快，这也许有点令人恐惧。

互联性在近年的巨大进展中扮演着重大角色。20世纪60年代以来，网络就是计算的一部分，但只有当人们开始将其个人计算机（及其处理能力的重心）与其他这类计算机连接起

来构成Internet之类的结构时，它才成为可能。遗憾的是，大多数网络的特征之一是，随着网络规模的膨胀，网络不同部分之间的交互常常导致反馈和其他意外现象，这使得建立紧密集成的系统更加困难。

数据+环境=信息

数据不是信息。数据是源材料，是形成信息的输入，但是，一般地，数据存在于与其产生的信息不同的环境中。例如，考虑一个监测某个特定地点（如机场）的湿度、温度、风速和气压等数据的小气象站。该气象站的任务是将数据定期传输给一个中央系统，把新获取的数据（如地点和时间）添加到基本记录中。

机场的空中交通管制系统将把该信息传送给飞机，帮助它们采取预防措施应付坏天气。新闻台气象员收集这些数据以及本地区其他气象站的类似数据，并显示在地区气象图上。气象学家利用该数据及更广区域的其他数据，准备一份给定地区未来几天天气如何演化的短期预报。气候学家则将该数据作为样本与其他数据结合在一起，以建立几年或几十年气象模式的长期模型。

这些人虽使用同一数据（单独或与其他数据联合使用），但他们却在不同环境内对其进行处理。即使所有系统技术上都能透明地通信，但它们仍然需要简便地转换数据环境的某些途径。目前，这种转换大多通过大量程序员来解决，因而代价昂贵（从Web开发人员的征聘广告数来看，程序员远远供不应求）。

上述天气例子说明，没有任何基本框架且没有环境的数据不过是一堆白噪音。数据只有置入某种形式的环境中，才能从中检索信息。可以把这种意义上的环境看成一个函数：作用于原始数据，过滤无关数据，返回相关的信息。对空中交通管制系统，当前风速是一个很重要的值，所以处理（环境）量相当小。另一方面，对气候学家而言，2000年3月3日下午2时奥林匹亚机场的风速无关紧要，除非它是一个大型复杂数据样本中的一点。

Internet的诞生

随着网络规模和数量的增长，这些网络产生的数据量也在增长，而且早就超出了人们能够有效管理的临界点。1997年夏，电子形式的数据量就已经超过了有史以来所有书面著作的总字数。到本书出版时（当然，也是书面形式），估计该数量将达到所有人类文献总字数的四倍，也就是说，从那以后，每18个月翻一番。

显然，Internet是世界上最大的网络，所以，许多信息超载问题首先在此网络出现就毫不奇怪了。Internet最明显（尽管不是惟一）的部分是World Wide Web（万维网），一个相互连接的HTML页面、服务器页面和类似资源组成的系统。尽管历史学家能够指出Internet中具有历史影响的可能事件的数量，但World Wide Web本质上是一个人1990年的发明，此人就是当时在瑞士日内瓦的CERN高能实验室工作的Tim Berners-Lee，他现在是World Wide Web联合会的主要成员之一。

说明： World Wide Web联合会（World Wide Web Consortium）也叫W3C，它是目前Internet上使用的大多数信息标准的最终仲裁者。在W3C Web网站上可找到最近的XML和HTML标准：http://www.w3.org。W3C在本书中具有重要地位。

Tim Berners-Lee创建基本Web协议——Hypertext Transfer Protocol（HTTP，超文本传输协议）和HyperText Markup Language（HTML，超文本标记语言）——时，他并未想到它们日后会风靡世界。当时，CERN的几位科学家需要一种以便于该组织内其他科学家引用并使科学家们能够引用彼此的超文本页面的方式来存储和传输文件，Tim Berners-Lee面对这种需求开始考虑解决办法。

虽然在此之前几年，超文本系统就已存在，但它们本质上是专用的。在Berners-Lee的设计中，主要区别是，他把文件浏览器（第一个浏览器）作为一个低带宽应用程序散发，并使协议和软件均可免费获得。不消说，关于这个应用程序的消息（及程序本身）传播得飞快，尤其是在HTML和HTTP的结合使其对教育机构产生重大影响以后。

HTML与XML的历史关系

HTML的历史已经家喻户晓，但值得强调它与XML相关的一个方面：HTML最初设计成一种传送和显示物理学摘要的方法。CERN的大多数摘要利用标准通用标记语言（Standard Generalized Markup Language, SGML）作为描述语言，主要用于存储论文、报告、产品说明信息和其他有关文档。

SGML可追溯到20世纪60年代。当时，IBM的Charles Goldfarb、Edward Mosher和Raymond Lorie为了帮助组织公司开始产生的大量文档而开发了SGML的前身——通用标记语言（Generalized Markup Language，GML）。到1978年，美国国家标准学会（American National Standards Institute，ANSI）采用GML的基础部分，制定了一份国家标准，叫做GCA 101-1983。六年后，由于GCA在文档管理上取得很大成功，国际标准组织（International Standards Organization，ISO）开始制定其全球版，这就是1986年发布的《信息处理——文本和办公系统——标准通用标记语言（Information Processing-Text and Office Systems-Standard Generalized Markup Language，SGML），ISO 8879:1986》。

Berners-Lee创建了HTML作为SGML的一个实例；他使用SGML奠定的规则建立书写摘要的标注。标题结构和HTML最早版本的类似元素在很多情况下都正常，但诸如CITE之类的标注至少提醒人们，Berners-Lee建立的结构从来就未打算超出创建摘要的目的。

然而，还是突破了该语言最初的宗旨。到20世纪90年代初，HTTP服务器排出的HTML页面风靡研究机构和大学。随着年轻程序员开始摆弄HTTP服务器和HTML浏览器，该语言开始以各种方式流传开来。当时，最成功的HTTP浏览器之一是Mosaic（马赛克），它由刚开放不久的设在依利诺斯大学的国家超级计算应用中心（National Center for Supercomputing Applications，NCSA）的Marc Andreesen等人创作。

> **说明**：作者注：NCSA开放时，我是依利诺斯大学的学生，我怀念与Andreesen（他已成为一位亿万富翁）一道工作的六个月。

Mosaic给Web戴上了图形外貌。在最初的HTML规范中，图像引用是指向图形文档的一个超文本链接，在另一个浏览器应用中可以打开该图像引用。有了Mosaic，就可以将图形图像直接嵌入文档中，本质上是把过去干巴巴的物理文件导航工具转变为大众交流媒介。Mosaic成为World Wide Web的流行应用程序，而Silicon Graphics公司创立者James Clark更与Andreesen合作创立了Netscape（网景）公司。

浏览器之争扼杀了HTML吗？

再回到环境问题。正如前面指出的那样，HTML的本来目的是描述科学摘要。它具有非常明确的结构，标题（如<h1>、<h2>和<h3>）反映了主标题和次标题信息，段落标注（<p>）包含信息段，引述标注指出文档引述，等等。如果你建立了这样的文档，那么你有理由期望程序装载该文档、分析它并产生非常符合你的要求的摘要。换句话说，早期HTML具有相当明确定义的环境。

图形表示的魅力及Netscape浏览器市场的飞速成长产生了一个问题。大多数人都不是SGML专家（谢天谢地）。那些开始摆弄Web页面的人凭感觉设计页面，根本不知道这个语言为什么具有这种结构。图形元素不是因为它是带有信息的元素而是越来越多地作为导航按钮、图像地图及相关组件而开始引入文档中。这样一来，Netscape就降低了进入的门槛，使得HTML分析器对马虎代码极其宽容——打开段落标注是隐含的、图像标注和规则不必关闭、重点（<EM>）和加强标注（<S>）分别成为粗体（<B>）和斜体（<I>）。

换句话说，HTML开始从一个基于环境的标记语言蜕变为一种排印标记语言。这并不完全出人意料。排印语言强调视觉表现，人们创作Web页面的理由是制作视觉显示，而不是引用学术摘要。例如，欲为客户制作醒目显示的广告代理商的Web制作程序就不需要摘要，他需要的是一种转换方法，转换以前用QuarkXPress和Aldus （现在是Adobe）PageMaker之类程序设计的精确的版面控制。

20世纪90年代中期的浏览器之战加剧了这个问题。由于未及时涉足Internet而饱受批评的比尔·盖兹选择1995年12月7日（此日是日军袭击珍珠港纪念日——译注）发出对占统治地位的Internet浏览器公司Netscape的挑战书，他宣布决心使微软成为以Internet为中心的公司。（他们是否已经成功仍见仁见智。）在随后四年中，微软和网景为争夺浏览器市场份额和HTTP服务器市场控制权而激烈撕杀。最后，网景败下阵来。1999年，America Online（美国在线）和Sun Microsystems（太阳微系统公司）联合收购了该公司，并将其改名为iPlanet，从而瓜分了战利品。

然而，这场战争的主要牺牲品之一是环境。几乎每隔几天就有向W3C提交的新标注建议，它们都有诸如blink（网景）标注和marquee（微软）标注之类旨在吸引用户的毫无用处的元素。

但是，如果blink标注无甚用处，那么其他标注就使最初文档语言中那点环境更少得可怜。例如，对表格数据，TABLE是语义上空洞的容器。一个单元跨多列或多行的能力意味着可能不再依赖基本表格标题信息来实际决定一列的含义。而且，网景与微软模型中表格之间的差异意味着几乎不可能将某些环境赋予一个表格，即便你想这么作。

FRAME是具有不佳环境结果的另一个HTML元素。利用帧来理解Web页面的含义实际上是不可能的。它还使Web页面中的导航复杂得多，对第三版HTML标准支持程度的不同使得针对帧的统一编程成为Web开发人员的噩梦。

但是，也许font标注是使HTML文档在环境上丰富起来的起点。font标注使你能规定跨度大小、字体名和颜色。如果你愿意，可以创建一个font标注，使得段落文本与h1文本相同（反之亦然）。尽管原先的意图相当好（把表现层与环境层分开），但大多数人和许多应用却利用font标注在设计Web页面格式时偷懒，而对程序结构不动脑筋。

当然，微软和网景参与的相互竞争的标准使所有这些元素情况更糟。两家公司都试图创建能给他们带来浏览器市场竞争优势的特征，而这些对HTML的增订意味着对两个主要浏览器的设计一般都要编写两套代码（或更多，如果新版本取代旧标准）。解决这个问题的最好方法最终可能归结为依赖Macromedia的Shockwave之类的专用解决方案，即保证两个系统中Web体验相同，而且搜索引擎和检索者基本上察觉不到的应用程序。

表现层减弱

到1996年，大多数Web开发人员都明显感到HTML受到破坏。浏览器之间不兼容、标注强调外在表现而非内容以及第三方解决方案的兴起都在一定程度上削弱了标准，简单地增加更多标注的技术似乎不再那么灵光了。

大约就在这个时候，W3C召集一个工作组研究在Web中实现一种样式表语言的可行性。样式表在页面发布领域已经存在一段时间了。本质上，利用样式表，你可以将一个属性集映射为一个已命名样式，然后将该样式应用于选出的文本（或页面上的其他元素）。

考虑过几种不同方法。首先考虑的方法之一是采用SGML的文档样式语义和规范语言（Document Style Semantics and Specification Language，DSSSL），它是一种转换语言，基本上处理标注并创建根据标注产生的格式化输出。DSSSL非常丰富而稳健，能够实际处理指令和文档的其他基本特征，但它也相当复杂，因此Web新手编写DSSSL代码可能很困难。

经过一段时间，W3C成员一致同意转而使用层叠样式表（Cascading Style Sheets，CSS）。在CSS中，每个样式由一条规则表示，而规则则由零个或多个组合成一个单元的属性组成。规则可以应用于单个HTML元素，也可以作为一个类（class）应用于特定标注。虽然CSS不如DSSSL稳健，但它的使用要简便得多，而且更适合微软当时开始推出的文档对象模型。

1996年12月，CSS1模型得到批准，这标志着动态HTML的开始。通过改变CSS属性而能够更改的HTML代码即所谓动态HTML。1998年5月，CSS的第二个草案（CSS2）获批准，这个草案包括媒体规范和国际化。目前正在考虑CSS的第三个草案（CSS3），该草案将考察性能扩展，可伸缩向量图形（Scalable Vector Graphics，SVG）以及把输入项特征结合进CSS2模型中。

说明：第2章“建立XML文档对象模型”将详细介绍层叠样式表。

关于CSS当前发展的详情，请访问Web网站：http://www.w3.org/Style/css。

XML的兴起

寻找更好的样式表语言透露出的最重要的信息也许是，Web界许多专家意识到，可能已经到重新考察使用HTML的原因的时候了。正如前文所述，HTML基本上是一种处理摘要的语言，尽管到1995年它已经发展成一种编辑语言。该语言对于编辑内容——新闻报道、文章、论文——相当胜任，但对要求在线出售产品的企业、把Internet看成一种为客户获取重要信息的途径的公司或多媒体网站，它似乎越来越力不从心。

到1995年，HTML中基于标注和属性的表达系统已成为标准记法，越来越多的公司推出自己专用的HTML扩展，虽然这些扩展主要是为服务器端代码创作而提出的。Allaire的ColdFusion是首批为建立Web页面而创建相容的专用标注集的扩展之一，微软则使用标注和百分号为其MSN系统实现创建语言。

然而，由于客户机浏览器不理解如何解释标注，这种系统仅在服务器上工作很出色。支持SGML的人士建议干脆在Web上使用SGML，但这种特殊方法有很多问题。虽然SGML本身不是专用语言，但它无疑过于复杂：语言规范达几百页，元语言的规则和例外令人晕头转向。

虽然这种复杂性对复杂的公司应用是必要的，但基于Web的解决方案要求SGML版本必须达到：

- 简洁
- 便于使用
- 例外尽可能少
- 可扩展（易于修改）

在这些要求中，SGML只满足可扩展标准。然而，鉴于HTML是一个SGML实例，创建一种体现SGML的最佳特征但表达简洁（HTML最突出的特征之一）的新语言似乎是合理的。1997年，W3C成立了一个Web和SGML专家工作组来尝试解决这个难题，并在1998年2月，接受可扩展标记语言（Extensible Markup Language，XML）第一版提案为推荐标准。

提示： XML第一版规范（称为WC3 REC-xml-19980210）可在以下网址找到：http://www.w3.org/TR/REC-xml。

基本词汇一瞥

尽管出于若干理由，典型的XML结构看起来很像被古怪地命名的HTML文件，但今天实际使用的大多数XML文档结构性相当高。一个简单例子是程序清单1.1那样的采购订单。

程序清单1.1 Acme Rockets的采购订单，一个XML文档

```
<?xml version="1.0"?>
<purchaseOrder type="1125" processed="false">
   <header>
      <orderFrom>Wiley E. Coyote</orderFrom>
      <orderTo>ACME Rocket Company</orderTo>
      <address>
         <street>1105 N. Sonoma Way</street>
         <city>Death Valley</city>
         <state>Arizona</state>
         </country>
      </address>
   </header>
   <body>
      <!-- The last rockets exploded prematurely. Please do
      watch your quality control. - WEC -->
```

```
        <lineItem>
            <name>Mark X Rocket</name>
            <description><![CDATA[Big, powerful red rockets,
    suitable for chasing highly mobile desert birds.]]></description>
            <number>24</number>
            <filter>if &gt; 15</filter>
            <unit>Individual</unit>
            <price>3.25</price>
            <priceUnit>USD</priceUnit>
            <discount>22.15</discount>
            <priceUnit>USD</priceUnit>
        </lineItem>
    </body>
</purchaseOrder>
```

这个例子虽然简单，但它显示了独立XML文档的大多数要素（独立的意思是说，文档中没有文档类型定义（Document Type Definitions，DTDs）或模式信息，两者均在第6章“XML模式”中详细介绍）。前例中，七类基本对象组成该文档：处理指令标注、元素标注、属性、文本、CDATA段、实体和注释，它们的特点详见以下几小节。

处理指令（PIs）

处理指令由打开和关闭问号标注括起来，如下例所示：

```
<? This is a processing instruction ?>
```

处理指令负责提供XML结构正常范围之外的信息，供分析器或第三方程序使用。例如，可以使用处理指令规定输出XML数据时使用哪些显示设置。正式XML结构应以<?xml version=“1.0”?> PI开始，以指出这是一个XML文档，尽管如果不包括这一首行，大多数分析器也不会产生错误。

元素

元素是由一对尖括号括起来的一个单词标注以及零个或多个名/值对（称为属性），如下例所示：

```
<purchaseOrder type="1125" processed="false">
```

关闭标注类似于HTML标注的格式，终止标注由</串（而非<）开始，如下所示：

```
</purchaseOrder>
```

在HTML中，有些元素不必明确地终止，如
标注。另一方面，在XML中，如果一个元素不包含任何文本或其他元素，仍须终止它。然而，可以使用这种元素的缩略形式，其中斜杠移到标注的末尾。在这种情况下，无需明确地关闭标注，就像下行所示：

```
<country/>
```

属性

属性是与某个元素相关联的名/值对。例如，在purchaseOrder标注中，type和processed都是属性名，如下例所示：

```
<purchaseOrder type="1125" processed="false">
```

组成属性与元素的应是什么，这个问题已引起许多服务器和新闻组的激烈争论，本章后面还将深入探讨这个问题。一般地说，元素增加处理的灵活性（但可能以速度为代价），属性易于把握但不灵活。属性不能包括分行符或XML符号（如<或&），许多分析器对属性的长度设置了上限（一般是256个字符）。

注释

XML结构中的注释是为解释或编码目的添加的说明，由一个惊叹号和两个短划线组成，如下例所示：

```
<!-- This is a single line note -->
```

注释可能包括空白（如字符间隔、分行符、跳格符及相关元素），但不能包含嵌套注释。换句话说，

```
<!-- This is a <!-- single --> line note -->
```

将产生错误，因为single后面的关闭括号只能被看成是注释的终止标注。

警告：不要将代码信息置于注释内。出于效率考虑，很多分析器实际上将注释信息从XML文档中剥离出来。使用PI。

文本

XML中的文本可以是任何统一代码（Unicode）字符（即遵守ISO统一代码标准的任何16位字符编码），但有几个重要例外。尽管可以将XML标注嵌套在文本内，但XML把这些标注当成子元素（见下节“CDATA段”）。因此，尽管下述加粗的字符被认为是文本：

```
<name>Mark X Rocket</name>
```

但只有Mark和Rocket被当成文本元素：

```
<name>Mark <X/>Rocket</name>
```

虽然不同的分析器以不同方式处理文本，但标准行为（以及微软分析器表现的行为）是把一个元素的文本定义为它的所有子元素的文本集。所以，下例中元素name的文本由firstName以及lastName的文本构成（例如，Joe Smith）：

```
<name>
   <firstName>Joe</firstName>
   <lastName>Smith</lastName>
</name>
```

元素文本的缺省行为是在检索文本时去掉多余的空白空间（如跳格、分行符、前导和末尾空格）。可以使用模式改变这种缺省行为，而保留某些或所有空白空间信息。