XSLT
开发人员指南

使用XSLT设计和构建复杂的数据驱动应用程序

将XML文档转换为灵活的、用于其他用途的数据

增强企业级应用程序的可维护性

Chris von See
Nitin Keskar 著
英 宇 译

清华大学出版社
http://www.tup.tsinghua.edu.cn
McGraw Hill

XSLT 开发人员指南

Chris von See
Nitin Keskar 著

英 宇 译

清华大学出版社

(京) 新登字 158 号
北京市版权局著作权合同登记号：01-2002-3181

内 容 简 介

XSLT 是 XSL(可扩展样式表语言)的派生产物，用来实现不同结构的 XML 文档之间的转换。本书首先介绍了 XSLT 的基本知识，接着全面深入地探讨了如何利用 XSLT 设计和构建复杂的数据驱动应用程序。主要涉及的内容包括：使用 Xpath 表达式定位数据、使用模板规则、创建 XSLT 结果树、使用 XSLT 中的变量和参数以及创建样式表输出等。最后还介绍了一些示例、编程工具及技术，以帮助读者更好地理解并运用 XSLT。

本书适用于希望了解和使用 XSLT 的开发人员和管理人员。

Chris von See, Nitin Keskar: XSLT Developer's Guide
EISBN：0-07-219408-1

书　　名：XSLT 开发人员指南
作　　者：Chris von See Nitin Keskar 著　英宇　译
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：徐燕萍
封面设计：康博
版式设计：康博
印 刷 者：北京昌平环球印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　印张：16.25　字数：416 千字
版　　次：2002 年 9 月第 1 版　2002 年 9 月第 1 次印刷
书　　号：ISBN 7-302-05825-3/TP・3449
印　　数：0001～4000
定　　价：32.00 元

前　言

回顾 XML 的急速普及，可以看到 XML 这样一个表面上非常简单的技术对计算机软件、数据库/数据管理和 B2B 通信世界的广泛适用性和深远影响是令人惊讶的。文档管理行业人员目睹了 XML 从 SGML 中的演化过程，他们知道基于标记的可扩展标记语言可以提供强大的功能；这种语言可以为构建数据提供一种高度结构化而且易于处理的方式，它不仅仅只是位和字节的不可识别流，而且还是一种语言，能够存储关于数据意义的信息。

自从 1998 年 2 月 XML 发布以来，整个范围的相关标准迅速地创建起来，而且均带有与 XML 相联系的名字：XLink、XPath、XPointer、XSL、XML Query、XFragment、XML Information Set 和其他许多标准。有些标准，例如 XPath(XML 路径语言)，自身或者作为其他技术和标准的组成部分都证明了具有非常强大的功能；其他标准仍然没有发挥它们全部的潜能，甚至永远都不会。

用于转换的可扩展样式表语言，或者称为 XSLT，是可扩展样式表语言(XSL)的产物，用于实现不同结构的 XML 文档之间的转换。在其第一个版本中已经证明了 XSLT 对于某些应用程序特别有用的，虽然在理解和高效使用上仍然有些困难。XSLT 是仅适于某些特定类型应用程序的一种技术吗？XSLT 适合干什么呢？怎样才能最好地使用 XSLT 呢？如何将 XSLT 集成到其他的应用程序中以创建更大的系统呢？

本书读者对象

本书主要适用于那些希望了解 XSLT 是什么，可以利用 XSLT 来做什么，以及如何用 XSLT 来解决商业问题的应用程序开发者、系统设计师和 IT 管理人员。尽管越来越多的技术主题需要通晓 XML 概念、HTML 和利用 Java/C++程序设计，但是即使您从未接触过 XML，想获得对 XSLT 的一般了解并非难事。如果您不熟悉 XML，我们希望第 1 章能够为您提供足够的有关如何开始 XSLT 编程的背景资料。

本书涵盖内容

既然 XML 和 XSLT 间关系密切，在开始学习 XSLT 编程前，能够认识到 XML 的基础知识是至关重要的。第 1 章，“XML 简介”为读者了解本书其余部分提供了基础，其中介绍了 XML 主要的基础知识，用简洁的术语描述了这一强大的技术。在这个背景之上，第 2 章，“XSLT 简介”根据 XSLT 的发展史与创建 XSLT 的原因，讨论了 XSLT 如何工作，并描述了能够在如今的软件库中创建一个强大工具所必需的关键特性。

第 3 章，“XML 和 XSLT 应用程序”，详细地研究了 XSLT 的应用程序。特别地，这一章讨论了 XML 最适合的企业级应用程序的类型、XSLT 对 XML 的补充以及在这些应用程序中的流线型应用程序开发。这一章还提供了一些在现实生活中使用 XSLT 的例子；该章的最后，学习完如何使用该工具后，开始学习如何开发自己的 XSLT 应用程序。

第 4 章 “XSLT 样式表的结构”，讨论了 XSLT 样式表的结构。不同于常规的编程工具，

XSLT 并未用作一种面向过程的语言，更恰当地说，它依靠调用模板的结构指出 XML 文档的哪些部分需要被处理。理解如何构建样式表以及如何通过 XSLT 使用模板，这是正确使用该工具的关键。

第 5 章“表达式”介绍了强大的 XML 路径语言(XPath)，它是 XML 标准集中的一个关键部分，在使用模板的处理过程中，XSLT 利用 XPath 语言在输入 XML 文档中选择节点。XPath 非常强大，但是强大是以复杂性作为代价的。我们一步步地讲解了这种路径语言，到本章结束为止，读者应该能够对 XSLT 样式表模板中的 XPath 表达式的使用有深刻的理解。

在第 6 章“模板规则”中，我们讨论了一些影响 XSLT 模板工作方式的规则。有些规则被构建到 XSLT 自身之中，而其他的则主要依赖于样式表中模板之间的关系。通过有效地使用(并操纵)这些规则，读者可以执行各种有趣且有益的 XSLT 特性。

在第 7 章“创建 XSLT 结果树”中，读者将了解到如何读入 XML 输入文档，并利用样式表的各部分处理输入文档中特定的元素。本章为读者介绍了 XSLT 所提供的实用工具，可以从样式表中生成静态和动态的输出，利用输入文档中的数据决定应该生成哪一种输出，并控制模板中的控制流。使用这些工具，可以对输入 XML 文档实现非常强大的转换，执行广泛的重组，以及结合使用 XSLT 和 XPath 函数进行数据处理。

第 8 章“变量和参数”介绍了 XSLT 中变量和参数的概念。变量和参数可以帮助开发普通的样式表，在样式表被调用时，变量和参数可以通过提供附加的信息定制具体的处理过程，它们允许在模板间共享数据，从而结合其他的 XSLT 功能。本章还描述了 XSLT 中的变量和参数与其他编程语言中变量和参数的区别。

第 9 章“创建样式表输出”介绍了如何通过 XSLT 样式表获得 XML、HTML 和输出文本。XSLT 提供了内置的函数，可以更加容易地控制输出文档的特性，例如文档的编码，并为自动生成编号和错误消息提供了工具。

为了能进行更加复杂的应用程序的开发，XSLT 为排序数据、通过属性和数据交叉引用元素、利用单个样式表处理多个文档等过程提供了工具。第 10 章“XSL/XSLT 高效编程”讨论了这些特性，并且进一步详细介绍了何时以及如何创建这些高级函数。

第 11 章“XSLT 示例”给出了一些现实世界中如何使用 XSLT 的例子，同时给出源代码。这一章串联了本书的所有其他各章，显示了如何集中利用 XSLT 的不同特性对目标 XML 文档执行强大的转换，根据其他样式表生成样式表等。

由于 XML 在企业级应用程序中的广泛适用性，我们认为讨论 XSLT 在这些环境中的相关内容有着重要意义。第 12 章“企业级开发中的 XSLT”更详细地解释了如何以及为什么将 XSLT 用于企业级应用程序。这一章同样讨论了现实生活中有关 XSLT 开发的几个关键问题，并提供了问题清单，以便于在开发复杂的 XSLT 项目前仔细检查。

第 13 章“编程工具和技术”讨论了目前可用的几种 XSLT 工具。尽管很多商业供应商支持 XML，但最流行的 XSLT 工具却在公开社区中。本章不仅给出了最流行的工具，也讨论了如何将这些工具与使用 Java、C++和 Perl 开发的应用程序相结合，并提供了可以找到这些内容和其他编程语言更多信息的网络资源。

既然 XSLT 1.0 是当前的 W3C 规范，本书中讨论的示例均满足这一点。然而 XSLT 和 XPath 版本 2.0 的工作草案最近已经发布了，我们期待着不久以后这一规范可以最终定案。基于此点，

附录 A 为 XSLT2 和 XPath2 提供了一个简短的评述。

附录 B 对 W3C 工作组(WG)作了一个简单的回顾，它们开发了 XSL/XSLT、XML Protocol、HTML、DOM、HTTP、密码学以及其他与 Web 技术相关的规范和推荐。

附录 C 为找到更多有关 XSL 和 XSLT 的信息提供了一个资源列表。这些资源包括书籍和一些有益的 Web 站点。

最后，术语表提供了关于 XSLT 和 XML 的术语定义。

如何阅读本书

本书假定读者至少有一定的 XML 基础，但是不需要了解 XSLT 的任何知识。如果您并不了解 XML，或者认为在开始学习 XSLT 前需要对相关内容进行复习，我们建议您在接触到本书更多技术之前，从第 1 章“XML 简介”读起。

由于后面章节中的技术材料广泛地构建于前面各章基础之上，我们建议读者能够循序渐进地阅读本书，虽然跳到您目前最感兴趣的工具和技术的章节或许更有效些。如果您只是一个 XSLT 的初学者，可以考虑从第 2 章对 XSLT 的介绍和第 13 章对 XSLT 处理过程的描述开始。如果您对 XSLT 已经相当地了解，想寻找 XSLT 所提供的更详细的功能，您可以查阅第 2 章～第 7 章，然后跳到第 11 章阅读对具体示例的讨论。如果您希望在学习 XSLT 语法之前了解 XML 和 XSLT 可以做什么，那么，第 1 章和第 12 章中的讨论是非常有益的。

目　　录

第1章 XML 简 介

本章内容包括：

- XML 历史简介
- 剖析 XML 文档
- 国际化 XML
- 作为元语言的 XML
- 与 XML 相关的 W3C 标准
- 小结

在本章中，我们会讨论 XSLT 语言的背景，该语言是由万维网联盟(W3C)制定的一个标准。XSLT 代表一种用于转换的可扩展样式表语言，它设计的目的是将 XML 源文件的树形结构转换成您所需要的特定文档格式。事实上，XSLT 样式表本身也必须是一个良构的 XML 文档。所以，我们首先介绍一下 XML 发展历史和其他一些与万维网(WWW)相关的发展，这样来开始我们的讨论是很有必要的。接下来我们将探讨 XML 文档的结构，讨论良构和有效 XML 文档的概念，进而简要地讨论一下相关的 W3C 标准。

1.1 XML 历史简介

XML 的起源可以追溯到电子出版领域。在 20 世纪 60 年代末期，IBM 的一些研究员致力于定义一种可以有效地管理大量电子文档的语言。在 1969 年，Charles Goldfarb、Ed Losher 和 Ray Lorie 一起发明了通用标记语言(General Markup Language，GML)，它正是 XML 的前身。GML 可以帮助编辑、格式化、文档出版，并非常适用于文本检索子系统。于是 Goldfarb 继续深入研究了这项工作，并在 1974 年发明了更加完善有效的标准通用标记语言(Standard Generalized Markup Language，SGML)。接着，在 1980 年美国国家标准协会(ANSI)委员会起草了 SGML 的首稿。

接下来，1983 年发布了最初的标准，并于 1985 年由欧共体发布了第一版得到业界认可的标准。当 SGML 于 1986 年被定为一项 ISO 标准(ISO 8879:1986)以后，它就被进一步合法化了。

一个有趣的花絮：最终的 SGML 标准就是使用 SGML 发布的。用来发布的系统是由 Anders Berglund 在著名的高能核物理实验室 Européen pour la Recherche Nucléaire(CERN)开发的，该实验室现在已经发展为欧洲粒子物理实验室。

SGML 被证明是一个有效和功能强大的电子发布信息标准，并被美国一些核心政府组织采用，比如国防部、美国国税局(IRS)，同时也被汽车业、电信业、宇航部、IBM 等一些大的公司所采用。不用多说，IBM 一直在支持 SGML 和 XML 的使用和发展。这些组织的共同特点是他

们规模大，相应地，他们就有大量的资源，并汇集这些资源来使用 SGML。小公司通常避免使用这种通用的语言，而是采用一些相对复杂的语言。

在 SGML 提供有力的和方便的方法用来标记电子文档的同时，也暴露了它复杂性的缺点，因而它只能吸引那些相对小范围的、特定的专业技术用户。为了能适应更加广泛的应用，急需一种在读取与编码方面很相似的语言。这就导致了超文本标记语言(HTML)的创建，它是由 Tim Berners-Lee 和在 CERN 工作的 Robert Cailliau 创建的。

CERN 作为一个重要的研究组织，拥有成千上万的科学家和大量的项目，他们使用成百上千的复杂计算机系统，运行着各种各样的程序。在 1980 年 Berners-Lee 作为 CERN 的一个技术顾问，开发了他的第一个超文本系统——Enquire，这个系统管理 CERN 内部的人员、项目和程序的相关数据。1984 年 Berners-Lee 正式在 CERN 任职并继续进行他的超文本研究工作。在 1989 年的 3 月，他提交了一个关于开发基于超文本的系统的提议，该系统具有开放的结构，并分布在通信网络的不同地点。当时 Cailliau 正在独立开发超文本系统，也参加了 Berners-Lee 的工作并帮助重写了原始的提议。于是，一个源于 SGML、简单的、基于标记的语言就开发出来了。这个 HTML 的前身的用户群很小，主要由一些欧洲的科学家构成，这些科学家利用这个语言来编辑并通过 EUNet 来共享电子文档，当时 EUNet 与 ARPANet 是连通的。

通过 Berners-Lee 和 Robert Cailliau 的共同协作和努力，诞生了第一个 Web 浏览器 WorldWideWeb，它是在一个 NeXT 计算机上开发的，并在 1990 年建立了第一个 Web 服务器 info.cern.ch。在 1991 年 Berners-Lee 慷慨地将这个 Web 服务器与全球计算团体共享了。接下来在 1992 年，由伯克利的加利福尼亚大学的 Pei Wei 开发和发布了另外一个 Unix 上的浏览器 ViolaWWW，ViolaWWW 提供了诸如图形和 applets 小程序这样高级的性能。

在 1992 年，工作在伊利诺斯州大学的超高速应用国际中心(NCSA)的 Joseph Hardin 和 Dave Thompson，下载了 ViolaWWW 浏览器并介绍给 NCSA 的软件设计团体。这次 Web 使用的展示引起了两名学生(Mark Andreessen 和 Eric Bina)的兴趣，他们在 1993 年开始了基于 XWindows 的 Web 浏览器的开发，并完全与计算团体共享。Andreessen 坚持不懈地与用户群保持联系并及时地获得这个不成熟的浏览器的反馈信息，以确保及时地修改错误和增强代码。Andreessen 接下来在 NCSA 领导一个软件小组开发了免费的公共 Web 浏览器 Mosaic。Mosaic 是基于 Berners-Lee 最初开发的浏览器构建的，它支持图形、声音以及影像。在 1994 年，NCSA 将浏览器的商业权利转让给了 Spyglass 公司，Spyglass 公司后来又将该技术授权给了 Netscape 和 Microsoft。

SGI 的创始人 Jim Clark 也参加了 Mark Andreessen 以及其他一些开发人员组成的 Mosaic 团体，并开始了商业性的交流，最终于 1994 年 5 月开发了一个商业化的 Web 浏览器。这个公司的名字后来改成了 Netscape。

同时，第一个 HTML 标准在 1994 年的第一次万维网(WWW)会议上产生并通过了。这与 Netscape 公司的产品发布一起带来了一个具有历史里程碑性质的事件：1994 年 12 月 Netscape 公司发布了第一个商业浏览器 Mozilla 1.0 版。

Mozilla 在一夜之间便声名显赫，为了对用户群的需求进行及时反馈，Netscape 公司在浏览器中增加了电子邮件和新闻组的功能。它的第一个版本是针对 Unix 操作系统的，Netscape 很快

就进军其他流行的操作系统，比如 Windows 和 Macintosh OS。在这段时间内，HTML 由于其固定的标记和易于理解的语法，被公认为比较简单也容易被初学者接受，并很快成为 Web 发布的标准。

HTML 除了简单这一特性，已经成为 Web 开发者的一个很好的工具，并能够满足每个人的任何疯狂的想像，这要感谢 Berners-Lee 的独创性的结构设计。使用这个静态的工具，几乎在一夜间就开发和发布了无数的 Web 站点。

然而近些年来，随着需求的变化和增强，HTML 的局限变得越来越明显了。HTML 是不可扩展的，也就是说如果开发人员需要的一些标记不存在于标准工具集中时，他们自己不能创建和定义新的标记。

HTML 着眼于在瘦客户端(也就是一个浏览器)管理表现方式，所以它不容易被重复利用。这个局限的主要结果就是 HTMLWeb 站点不容易升级，而且不能满足在这个 Web 站点上产生的一些新需求。简而言之，仅仅使用 HTML 来构建和管理复杂的、动态的 Web 站点对于 Web 站点管理员来说简直是一场恶梦。而且，HTML 自身仅可以提供一种内容和数据的显示风格，这违反了电子出版业的一条基本的指导原则：数据与表现分离。这些缺陷，加上一些其他的因素，导致了开发一种可扩展开放标准的需求，并能与 HTML 一样可进行移植。SGML 的设计完全符合这些要求。可是 SGML 的复杂性使得它只适用于 SGML 专家。这些因素的综合，便为 XML 和相关标准的出现创造了良好的氛围。

在 1996 年的早期，W3C 的 Dan Connolly 就预见了人们对于与 SGML 相关的标准的需求，虽然 W3C 并没有从事这项工作的相关资源。当时在 Sun 公司的 Jon Bosak 在 Novell 的一段时间内与 W3C 交流过关于这样一个开放标准的事宜。1996 年 3 月，Connolly 让 Bosak 在 W3C 组织和领导一个工作组，利用 Sun 公司的一些资源将 SGML 扩展到 Web 上。这个方法与 W3C 的其他一些工作组不同，其他的工作组都是由 W3C 内部的员工组织和启动的。Bosak 同意了这个提议，并与世界上最重要的 SGML 专家交流，包括在 WWW、SGML 和 ISO 会议的专家，探讨了他在接下来几周的工作。接下来的工作被命名为“Web SGML 行动”，当然，同期的一个 SGML/DSSSL 专家 James Clark 提出了另外一个名字：可扩展标记语言(XML)。您可能注意到虽然流行将 Extensible 写为 eXtensible，但是前者才是 XML 的全称中的正确单词，也是写入 W3C 规范中的单词。

基本的设计是在 1996 年的 8 月，在 Tim Bray、Jean Paoli 和 C. M Sperberg-McQueen 的带领下，在短短的 11 个星期内迅速完成的。这个规范的最初草稿是在 1996 年 11 月的 SGML96 年会上提出的。这个 W3C 工作草案在 1996 年 11 月发布，并称 XML 是 SGML 的极度简化版。XML 的建议稿大约在一年后即 1997 年发布，最终在 1998 年 2 月成为技术规范的 1.0 版本。于是，XML 在极短的时间内变成了一个众所周知的名词，并且它的发展没有半点放慢的趋势。

现在 XML 获得了很多软件业的大公司和商家的支持，像 Sun、Oracle、 Microsoft、SAP、IBM 和 Adobe。它已经变成了网上描述数据和内容的真正标准。在下面一节中您可以看到，XML 是很容易读写的，它是描述机器可读的结构化数据的强大工具，在构建一个包括 RDBMS、Java、HTML、XSL 和 XSLT 等多种技术的复杂 Web 应用中，XML 是非常有效的。

1.2 剖析 XML 文档

本节我们将讨论 XML 文档的结构，以及它们的基本组成部分。剖析 XML 文档的最好办法是浏览一个示例文档，然后讨论其中每一个部分。所以我们选择了下面的 XML 文档 **auth.xml**，它输出一个作者列表。

```
<?xml version="1.0"?>


<!-- data specifications begin-->
<All_authors>
  <Author AuthID="77">
<Name>
  <First_name>Samuel</First_name>
  <Last_name>Clemens</Last_name>
</Name>
<Address>
  <Address_line1>111 Mansion Boulevard</Address_line1>
  <Address_line2>Suite 100</Address_line2>
  <City>Chicago</City>
  <State>IL</State>
  <ZIP>31313</ZIP>
</Address>
</Author>

<Author AuthID="88">
...
  </Author>
</All_authors>
<!-- Data specifications end -->
```

下面几小节讨论了这个 XML 文档的各个组成部分。

1.2.1 XML 声明

XML 文档的开始，如<?xml version="1.0"?>，它简要地说明这是一个遵循 W3C 标准 1.0 规范的 XML 文档。您可以有选择性地增加 XML 所遵循的 ISO 规范，就像下面的代码行所表示的一样：

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

1.2.2 注释

在 XML 文档中任何需要的位置都可以在<!--和-->之间添加注释。

1.2.3　标记

就像 HTML 一样，XML 标记也是由限定符< >括起来的，比如<All_authors>。然而，它们也有一些区别，HTML 所使用的是一些固定的标记集，而在 XML 中您可以自由地为标记起名字，只要您不使用类似于单引号或双引号这样的特殊字符。第二个区别是 XML 中的标记是区分大小写的。第三个主要的区别是在 XML 中每一个标记都需要有结束标记。在下面的例子中，</First_name>标记就是结束标记：

```
<First_name>Samuel</First_name>
```

1.2.4　元素

元素是由起始标记、结束标记以及标记之间的内容（通常被称为“元素内容”）组成的。下面的代码展示了一个完整的 Name 元素，这个元素还包含几个其他的元素：

```
<Name>
  <First_name>
    Samuel
  </First_name>
  <Last_name>
    Clemens
  </Last_name>
</Name>
```

每一个 XML 文档都需要有一个而且只有一个包含所有其他数据的文档元素。这是数据描述中的第一个元素(或者叫最外层元素)，被称为文档元素或者根元素。在 XML 文件 auth.xml 中，<All_authors>是文档元素。

所有的元素都需要恰当的嵌套，也就是说除了文档元素外的每一个元素都必须完全包含在另外一个元素内。例如，由于元素 First_name 的起始标记包含在另外一个元素 Name 中，那就必须有一个相应的结束标记</Name>。下面的代码是无效的：

```
<Name>
  <First_name>
    Samuel
  <Last_name>
    Clemens
  </First_name>
  </Last_name>
</Name>
```

关于标记和元素的警告：不可以在 XML 文档中使用 HTML 标记，因为不能期望解析器能把它们作为 HTML 标记来处理。这是因为 XML 的解析器并没有假定某一个标记的具体含义，这个责任完全在于编程者自身。例如，如果在 XML 文档中使用<B></B>标记，解析器并不知道您想要让这个标记所包含的内容在输出的时候用粗体显示。