Internet 新技术丛书

HZ BOOKS

XSLT和XPath
——XML转换指南

XSLT and XPath: A Guide to XML Transformations

本书是成功使用XML所必需的
——Sharon Adler, W3C XSL工作组联合主席

(美) John Robert Gardner
Zarella L. Rendon 著

飞天工作组 译

机械工业出版社
China Machine Press

PH PTR

Pearson Education

Internet新技术丛书

XSLT 和 XPath

——XML转换指南

XSLT and XPath: A Guide to XML Transformations

John Robert Gardner
（美） Zarella L. Rendon 著
飞天工作组 译

机械工业出版社
China Machine Press
PH PTR Pearson Education
培生教育出版集团

本书系统地介绍了使用XSLT和XPath进行XML转换的方法。本书概念清晰，循序渐进，并结合大量实例，详细讨论了进行XML转换所涉及的样式表、元素、函数、表达式等的具体使用方法，还介绍了三种免费的XSLT解析器，并在书后附录中给出了典型案例及应用。

本书附赠的光盘中包含书中所有例子的源程序代码。

本书版权登记号：图字：01-2001-3873

图书在版编目（CIP）数据

XSLT和XPath——XML转换指南/（美）加德纳（Gardner, J. R.），（美）雷登（Rendon, Z. L.）著；飞天工作组译. -北京：机械工业出版社，2002.4

（Internet新技术丛书）

书名原文：XSLT and XPath: A Guide to XML Transformations

ISBN 7-111-09980-x

Ⅰ. X… Ⅱ. ① 加… ②雷… ③飞… 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014182号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：宋燕红 刘立卿

北京昌平奔腾印刷厂印刷·新华书店北京发行所发行

2002年4月第1版第1次印刷

787mm × 1092mm 1/16 · 20.25印张

印数：0 001-4 000册

定价：42.00元（附光盘）

译 者 序

XSLT是将XML转换为HTML和其他浏览器可解析形式的有效工具。本书系统地介绍了使用XSLT和XPath进行XML转换的方法。既有相关概念的阐述，又有具体方法的分析。作者力图通过本书，使读者对HTML、XML、XSLT、XPath等基本概念及其相互关系有比较清楚的理解，并在此基础上掌握XML转换的具体方法。

本书主体可以分为三部分：第一部分阐述了基本概念及其相互关系；第二部分结合大量实例，详细讨论了进行XML转换的方法和所涉及的样式表、元素、函数等的具体使用方法；第三部分介绍了三种免费的XSLT解析器的安装及使用方法。本书后面还有三个附录，给出了几个案例和典型应用。

本书对几个重要的概念及其相互关系阐述得很清晰，同时给出了大量的源代码实例，可以说是关于使用XSLT和XPath进行XML转换的比较全面的指南。本书适合想尽快了解和掌握XSLT及XML的初学者，也适合有一定基础，想进一步提高的程序开发人员、Web管理员等。

本书的翻译是在飞天工作组全体成员的共同努力下完成的，他们是葛新、刘实、蔡一兵、朱军、朱琳杰、马长斗、宋慧、林建国、诸志远、陈杰、张一鸣。由于XML和XSLT技术发展异常迅速，原书中出现许多新的概念和名词，工作组成员多方查找相关资料，尽力符合原意。但由于译者水平所限，书中定有很多不当之处，请读者批评指正。

2002年1月

前　言

你一定听说过XML，你的老板想要你把它应用到你的工作中，你该怎么办？

也许你用过HTML，而且知道标记是什么，知道标记与XML是有关联的，甚至你知道XML是什么和它能做什么。但你可能不知道的是，当XML识别文件的内容和向文本中增加结构时，它并不能告诉你怎样去处理这些内容，或者除了存储这些信息以外还能做些什么。这是件好事情，因为这意味着你的内容可以用在许多不同的场合。

一旦你将内容用XML标记过，你就可以用它去表示许多事情，但这里我们仅选择一些典型的或者说标准的应用来讨论。使用XSLT，你可以为XML加入样式，把它变成其他的XML，或者只是把它拆开，然后将它重新生成为不同的形式。

XSLT就像是藏在XML王冠之后的权力，它可以访问任一层次上的任一部分的XML数据，并可以重新使用它。可以毫不夸张地说，XSLT和它的同伴XPath就像是胶水和胶泥紧密地联系在一起，无论是在商用、学术研究，还是编程者个人兴趣开发上，它都能产生无穷的应用变化。掌握了XSLT，你不必害怕在数据或信息结构设计领域过早地落伍。

XSLT很容易使用。事实上，XSLT本身就是XML，XSLT就像一种可以说的语言，它具有语法、可供编程的丰富词汇；同时，它又很容易理解、学习和使用。

XSLT将为非编程者搭建一座桥梁，它提供了一种理解XML语法的简单途径，同时XSLT还拥有了丰富的脚本编写机制和进行文件导航的便利方法。

本书的宗旨是，使有着丰富编程经验的人，或者是刚刚学会标记语言的新手，都能够更方便地使用这套强有力的工具来维护、添加、更新和传递XML数据——无论这些数据是用在Web上、内部局域网上，还是B2B（商业到商业）商业模式中。

如果你希望不断提高控制信息的能力，本书将告诉你如何去做（当然还不仅如此）。实际上，到第1章结束时，你将能学会XML文档转换到HTML的基本方法，生成的HTML可以在任何一个Web浏览器上浏览。相继的几章将进一步巩固和加强这方面的基础知识，配合示例进行更详细的解释，同时对一些容易被误解的地方进行重点说明。

当你读这本书时，把你的计算机打开放在手边，花点时间安装一种XSLT的解析器（解析语言），然后一边看书一边跟着操作。在实践中学习是最好的方法，尤其对于学习XSLT和XPath。第13章将向你介绍如何安装CD中的软件，书中的每一个例子都可以在CD中的example目录中找到，并且是按章节的顺序组织的。

XSLT是一种激发人们创造力的语言，同时，它也会给你丰厚的回报。准备好享受这趟学习之旅吧，本书一定能带给你惊喜！

为什么我们要使用XSLT

浏览器能够表示的是HTML，而不是一般的XML标记。一旦你有了XML，你必须还要对它做些什么呢？你能用XML打印吗？你能把XML发送到Web上吗？你能浏览XML吗？是的，你能，

但是仅有XML是不够的。

XSLT让你能够将XML转换成HTML或另一种类型的XML，或者仅仅是简单的文字。仅需一点点的创造力和正确的XSLT知识，你就能熟练地将XML输出到实际应用所需的任何一种形式。

对于所有这一类XML转换的问题，XSLT提供了快速、简便的解决办法。但是使用XSLT也并非能无师自通。

"这本书，连同所使用的相应的工具，都是在实际应用中成功使用XML所必需的。"

——Sharon Adler，W3C XSL工作组联合主席

本书出版之时，XSLT的最新版本是1.0。W3C 的XSL工作组正在考虑在2.0版中加入一些额外的特性，包括支持XML模式、XML查询等等。

本书的读者对象

本书是为那些整日和电子数据打交道，想掌握XML转换方法却又不想经过学习编程语言的艰难过程的人准备的。如果你对使用SGML、XML甚至HTML感觉很顺手，你将会从这些通用的标记语法中获得巨大的收益。

有些人认为XSLT比较难，因为它不是一种结构化的编程语言。大部分的编程语言都具有结构化、简练的句法，而XSLT的句法是XML，它被设计成一种适合人们阅读习惯的和易于理解的句法。在你使用XSLT之前，最好能有一些标记语言方面的知识。

有些人可能发现XSLT较难使用，因为它不能对每一种情况都提供转换的解决方法。举个例子来说，你不能用XSLT去把文字转换成XML，这就需要另外进行处理。但是对于大部分天天使用XML转换的人来说，XSLT仍然是一种首选的工具。

本书的组织结构

这本书是按照如下方法组织的：首先建立一个基本概念，然后按章节逐步增加内容。首先在第1章介绍了XSLT的基本概念和简要概述了XML，在其后的章节中，随着创建样式表逐步增加功能的介绍，越复杂的问题越放在后面。

第1章用最简练的语言概括了XML和XSLT的基本情况。这一章对最基本的语法做了简要的总结，可以为对标记语言有不同层次了解的人提供一个总体的概念。

第2章介绍了样式表概念，这对于理解XSLT是一个至关重要的概念，同时也给出了关于样式表的通用的专有名词解释。

第3章增加了更多的相关概念，并提供了更详尽的解释和用法；同时也对第1章和第2章涉及到的模板有了更深入的研究。

第4章定义并解释了XPath的表达式和模式。

第5章包含了XPath的函数，这些函数对于掌握XSLT中的大多数元素的用法是至关重要的。

第6章介绍了生成新的XML元素及其属性的几种不同方法。

第7章讨论了如何引用和插入多样式表，也讨论了模板优先级问题。

第8章介绍了如何使用变量和参数。

第9章涉及了一些具有迭代性或条件受限的情况，以及从输入到输出复制XML所需的工具。

第10章给出了如何控制输出选项的一些细节，包括删除或保留空白域、产生出错信息等。

第11章介绍了XSLT的函数及相关的元素，包括用document( )函数插入外部的XML文件，用<xsl:key>使用键值(key)等。

第12章讨论了XSLT的扩展、解析器和Java，同时介绍了三种商用的XSLT解析器。

第13章介绍了三种免费的解析器——Xalan、Saxon和XT，并对它们的安装指令和外围组件进行了描述。

三个附录分别介绍本书的几个相关的主题、研究实例和相关的参考资料。

本书中的一些约定

本书中提到XML、XSLT和HTML元素时，总是对它们做出一定的标识。例如，<xsl:stylesheet>，总是用尖括号<和>括起来，并使用印刷字体。任何一种表达式或函数，比如count( )，也都用印刷字体。

每一个元素都有一个元素模型的定义，它可直接由XSLT来规范，如下所示。元素模型被组织成为一个带有选项的目录描述（作为一种注解）的XML元素，它都是由具有合法属性的起始标记、内容（作为一种注解）和结束标记（除非元素是空白）三部分组成。属性用黑体表示，它的值如果是有定义的内容类型，用斜体字表示；如果是字符型的，用引号括起来。内容中的元素可以是任意的，用a? 表示；也可以是任意的并可重复的，用a* 表示。

```
<!-- Category: instruction -->

<xsl:for-each

  select = node-set-expression>

  <!-- Content: (xsl:sort*, template) -->

</xsl:for-each>
```

函数原型从XPath和XSLT规范中直接获得，格式是这样的：首先是关键字Function，后面是冒号，跟着用斜体字表示目标返回类型，函数的名称用黑体表示，圆括号里的斜体字表示参数，如下所示：

```
Function: number sum (node-set)
```

版本

本书中所使用的语言的版本号如下：XSL转换（XSLT）是1.0版；XML路径语言（XPath）是1.0版；可扩展标记语言（XML）是1.0版。其他参考材料来自于XML REC-xml-names-19990114中的Namespaces。

在这本书中，James Clarks的 XT测试的版本是19991105，Michael Kay的Saxon用的版本是6.2.2。

目　录

第1章　一张XSLT样式表的剖析

- XML总览
- XSLT和XPath的介绍
- 节点
- 文档排序
- 从XML到HTML的转换

XSLT，又称可扩展的样式表转换语言，提供了一种转换和操作XML数据的机制。离开XML（extensible markup language）来讨论XSLT是没有意义的。XML是由W3C(World Wide Web Committee)制定的标准，它构成了信息互交换标准的基础。XML提供了信息构成的结构，XSLT与另一个相关标准XPath(XML路径语言)，提供了提取、重建和熟练使用XML中信息的手段。

XSLT具有与XML相同的交叉平台功能，因为它与XML具有相同的规则。如果你是一个在XML领域中有着丰富经验并精通标记技术的人，你将能很快地并最大限度利用XSLT这种独特的丰富语言。实际上，到这一章结束，你将能创建XSLT的样式表来完成基本的从XML到HTML的转换。

这一章提供了使用XML、XSLT和XPath的解释说明和分析。如果你有任何一种标记语言的先验知识都会大大加快你的学习进程。因此，本书将在下一小节中对相关知识做一个简要的回顾。

1.1　什么是标记

在这里不用再重述标记(Markup)出现的历史或者它具体是一个什么东西，重要的是对XML给出一个概念性的理解，本书不想具体提供对XML完整的句法和使用的解释，但是有些基本概念是值得回顾的。XML是一种标记语言，它的起源或者风格来源于SGML(标准可标记语言，Standard Generalized Markup Language)，大家可能对HTML更熟悉，它可在Web上表达标记的概念，它就是SGML一个普遍使用的例子。

标记由标志(tags)组成，它用来从文档的表达、样式中来描述和分离一个XML文档实例的内容，这些标志可以被认为是一对句柄，通过它，所有包含的文字、数据材料都可以被读取或识别出来。

我们称标志所包含的目标为元素(elements)。元素是一个XML文档中主要构成成分，它可以根据元素类型的名称识别出来，就像下面表示的包含在开始标记和最后标记之间的部分：

```
<para>This is a paragraph</para>
```

上式中的<para>和</para>为段落标记，用来把段落的内容从其他段落和XML文档中的元素(elements)分离开。注意到，这里我们已经标记了自己的元素类型名称(取代过去HTML中的旧的<p>标志)。XML允许创建自己的标志名称，可以无限制地增加比HTML可提供的更多的函数功能和标记的精度。作为另一个XML的特点，元素不但可以包含文字（像上面例子所示），还可以包

含其他元素，如：

```
<para>This is a <index>paragraph</index></para>
```

元素中包含元素引入了嵌入的概念，同时也定义了元素之间的关系，它可以直接进行寻址：根据一个元素在另一个元素中的位置和相互关系，就可以马上直观地表达出来。就像例子中“所有的元素索引在一个段落中”，嵌入的元素也可以再创建结构，这样就可以表示成一种树状的关系。

标记生长的树状图

当我们解释标记时，通常用树状图来描述概念、设计、处理过程、其他实体和非实体的观点。树根尤为特殊，在人类思想和哲学发展的历史长河中，它一直作为一种基础，或是起点。在XML中的根是相似的，从XML文档的根开始，引入了一个树状的结构，以它为基础可以进行精确的引导和参考。这种精度对于XML的使用是必不可少的。一个XML文档的接入和实体结构的使用和它作为树状结构的表达方式来看是不可分割的。

理解一个XML文档的树状结构对于这个文档来说是非常重要的。标记的树，特别是在XML中，是被高度地抽象化并按照标记语言的特征和风格进行转换。它是一种自顶向下的结构，就像下面图1-1中举的一本书的例子。

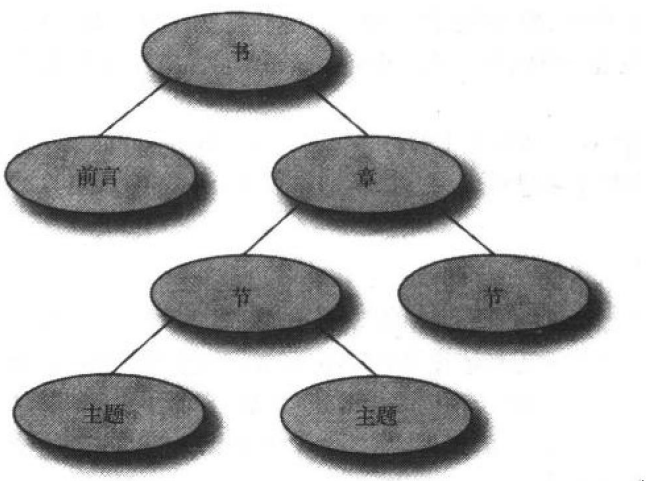

图1-1 一个基本XML文档的表达方式

XML有许多其他特征，足够写成其他数不清的书籍进行详述。由于我们主要为了讨论XSLT及它与XML的关系，这里我们将不会覆盖所有与XML相关的概念，仅讨论那些与使用XSLT直接相关的部分。

1.2 什么是XSLT

从本质上来说，XSLT就是XML，它使用相似的标记结构，使用“大于”和“小于”符号("<"和">"，例如，在<xsl: stylesheet>中)，XSLT的句法很容易识别。

把XSLT认为就是一个XML文档的实例有许多好处，当然，除了熟悉的标记结构，还需要明确以下几点：它们有着同样的句法；XSLT具有独立的平台，掌握了同样的基本技能就可以对它

从语法上进行分析。

另一种认为XSLT就是XML的好处就是它合乎文法的观点，允许不需要特定的DTD[⊖]就可以进行对XSLT样式表的结构进行处理。XSLT样式表合乎文法的重要性不仅如此——还包括当它初次被XSLT解析器读取时就可以被成功地解析；并且易于理解、调试或调整。

XSLT可用来将XML文档转换成其他XML文档，XSLT解析器从句法上分析输入的XML文档，包括XSLT样式表，处理从XSLT样式表中找到的指令，使用输入的XML文档中的元素。在处理XSLT指令的过程中，创建了结构化的XML输出，XSLT指令以XML的元素形式存在，用XML的属性去访问和处理XML输入文档中元素的内容。

XSLT的主要用途不是在于格式化，W3C组专门为格式化而独立制定的规范称为XSL[⊖]，通常称为XSL FO(格式目标，formatting objects)。如果需要的话，XSLT可以影响格式，举个例子，XSLT样式表可以被设计成输出HTML标志，并在浏览器中显示，但是这仅仅是它功能的细枝末节。

1.3 什么是XPath

XSLT极少离开XPath[⊜]而被单独讨论。XPath是一个来自W3C的独立的建议，它使用一种简单的路径语言来对XML文档的各部分进行寻址。尽管XPath也被其他W3C建议所引用，但极少有不涉及XPath而单独使用XSLT的情形。总之， XSLT提供一系列的操作和操作方法。而XPath保证了选择和寻址的准确度。

XSLT样式表

图1-1是“书”这个例子中元素的结构化层次表示图，它是开始理解XSLT样式如何工作的有效途径。图1-2表示同样的树状结构，不过这里它反映了XSLT样式表的一些基本成分。

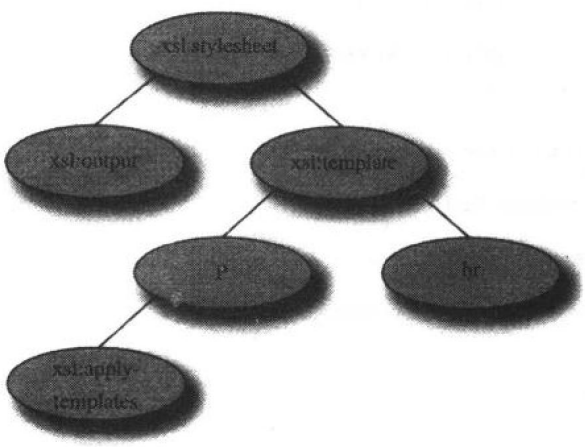

图1-2 一张基本XSLT样式表的树结构表示

⊖ XSLT没有一致的特定的文档类型定义(DTD)，但是基本的元素组是支持XSLT处理器在XSLT规范中用非标准的DTD来描述。

⊖ XSL的规范可以从http://www.w3.org/TR/xsl/网址中找到。

⊜ XSLT和XPath两者都是在1999年11月16日成为完整的推荐标准

本章中后面的内容将会对如何使用和解释这些元素提供更详细的叙述，这里所要强调的是XSLT就是XML，有着相同的整体结构。

1.4 XSLT样式表概念

根据样式表的结构和在样式表中被命名的元素，可以很容易地理解XSLT样式表的概念。它有着标记语言的显著特征，这些表明XSLT一直在努力地尝试使元素类型的名称和其他可能有的成分更符合人的阅读习惯。使用XSLT，这些都可以相当好地被完成，也使XSLT样式表变得更加容易学习和掌握。

让我们来比较一下一本书的XML树结构和一张XSLT样式表的树结构，见图1-3。

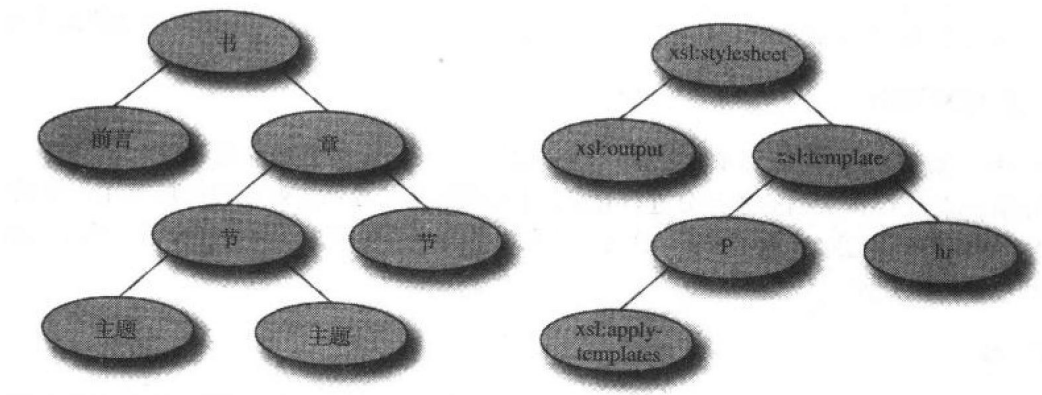

图1-3 比较XML树和XSLT树

如果解释右边XSLT样结构图作为一张样式表，它将看上去和例1-1相似。

例1-1：XSLT样式表作为一份XML文档

```
<?xml version="1.0"?>
<xsl:stylesheet
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      version="1.0">
      <xsl:output method="html" />

      <xsl:template match="topic">
            <p>
                  <xsl:apply-templates />
            </p>
            <hr/>
      </xsl:template>
</xsl:stylesheet>
```

这个例子表明了大部分XSLT样式表的基本成分，<xsl:stylesheet>元素包含了两个其他元素，一个是<xsl:output>，一个是<xsl:template>元素(有时也称作模板规则)。

HTML的<p>标志简单地发送相同的标志到输出，<hr>标记也如此。

在我们例子中另一个成分是<xsl:apply-templates>元素，它在<p>标志对的内部。

本例中，通过使用<book>，XML将产生如例1-2中所示的输出，。

例1-2：处理一个主题

INPUT:

```
<book>
      <intro></intro>
            <chapter>
                  <section>
                      <topic source="song">Xanadu</topic>
                      <topic>Topic 2</topic>
                  </section>
                  <section></section>
            </chapter>
</book>
```

OUTPUT:

```
<p>Xanadu</p>
<hr/>
<p>Topic 2</p>
<hr/>
```

注意例1-1中XSLT样式表的结构对这个输出的作用。<xsl:template>元素在输入的XML文档中发现了与<topic>相匹配的元素，它将被<xsl:template>元素的内容的取代——在这个例子中是 和<hr/>元素，然后<topic>的内容被发送到输出，它的内容被限定在 和 这对开始和结束标志之间，这些将由<xsl:apply-templates>元素来完成，该元素被包含在 标志范围之内。

同时也应注意到产生的输出并不是形式良好的XML，XSLT解析器可以产生XML文档，但不对输出的文档做句法上的分析。

使用XSLT把XML转换成HTML

让我们用一些简单的XSLT来把在XML文档中的元素转换成HTML。举个例子，我们考虑通常意义上的一年的情形，根据农业节气用播种、收获、季节、月份的自由观点进行细分[⊖]，例1-3表明了怎样用标记来描述一年。

例1-3：使用XML来标记一年

```
<?xml version="1.0"?>
<year>
      <planting>
            <season period="spring">
                  <month>March</month>
                  <month>April</month>
                  <month>May</month>
            </season>
            <season period="summer">
                  <month>June</month>
                  <month>July</month>
                  <month>August</month>
```

⊖ 当然，播种和收获是不相同的，季节也是不相同的，这些都是普遍存在的（随着全球气候的改变，你的气象经验也可能截然不同），所以这是一种通常的表达方法，主要基于关于北半球季节和气候（通过我们的Iowa朋友可能为我们承担一部分的任务）。

```
            </season>
        </planting>
        <harvest>
            <season period="fall">
                <month>September</month>
                <month>October</month>
                <month>November</month>
            </season>
            <season period="winter">
                <month>December</month>
                <month>January</month>
                <month>February</month>
            </season>
        </harvest>
</year>
```

假设需要解释我们的一年以便于在常规的Web浏览器上显示，将XSLT样式表转换成HTML是一个经常性的任务，我们将做这样一种简单的变换，用HTML非顺序列表格式(
)来显示<month>。

使用例1-4中的样式表，我们用HTML列表项标志(
)为输出文档创建一个简单的月份列表。

例1-4：无序列表的基本样式表

```
<?xml version="1.0"?>
<xsl:stylesheet
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      version="1.0">
<xsl:output method="html" />
<xsl:template match="year">
          <ul>
                <xsl:apply-templates />
          </ul>
</xsl:template>
<xsl:template match="month">
          <li>
                <xsl:apply-templates />
          </li>
</xsl:template>
</xsl:stylesheet>
```

再考虑本例中XML文档的XSLT样式表结构，如果把这些所需要的样式表成分放到以后去讨论。可以看出剩余的两条模板规则还是很简单的，第一个用于寻找和<year>(用match="year"属性) 匹配的规则并用无序的列表标志(
)取代它。<ul>元素是<xsl:template>元素的子元素，然后在<ul>元素范围之内是指令——<xsl:apply-templates>——去处理任何一个<year>的子元素。

<xsl:apply-templates>指令元素的基本功能是通知处理器为<year>的每一个子元素寻找一个匹配的<xsl:template>，它对每一个子元素及其后代进行循环寻址，直到处理完所有的后代。如果处理器发现匹配某一个元素的规则，它就会按照这个模板规则下的指令去处理这个节点。如果处理器没有发现匹配的规则，它就会继续工作下去，遍历所有的子孙直到它到达下一个节点。此时，