The XSL companion

XSL 编程指南

Neil Bradley 著
何健辉 吴益华 译

ADDISON-WESLEY

中国电力出版社
www.infopower.com.cn

The XSL companion

XSL 编程指南

Neil Bradley 著
何健辉 吴益华 译

中国电力出版社

内 容 提 要

可扩展样式表语言（XSL）是万维网联盟提出的标准族。本书全面介绍了 XSL 应用的各个层面，内容包括模板、样式表、HTML 输出、属性表达式、排序、编号、XSL 输出、文本格式、名字空间、CSS、XSLT 扩展等。

本书适合网站设计、开发、管理人员及大专院校师生阅读。

图书在版编目（CIP）数据

XSL 编程指南/（美）布来得利编著；怀石工作室译.-北京：中国电力出版社，2001.10

ISBN 7-5083-0800-X

Ⅰ.X… Ⅱ.①布…②怀… Ⅲ.可扩充语言，XSL-程序设计-指南 Ⅳ.TP312-62

中国版本图书馆 CIP 数据核字（2001）第 069172 号

北京版权局著作权登记号 图字 01-2001-4267

本书英文版原名：The XSL companion

中国电力出版社出版、发行

（北京三里河路 6 号 100044 http://www.infopower.com.cn）

实验小学印刷厂印刷

各地新华书店经售

2001 年 11 月第一版 2001 年 11 月北京第一次印刷

787 毫米×1092 毫米 16 开本 19.5 印张 430 千字

定价 34.00 元

前　言

本书覆盖了由万维网联盟（World Wide Web Consortium）提出的标准族。这些标准形成于一个 1997 年提出的、被称作 **XSL**（可扩展样式表语言，eXtensible Stylesheet Language）的样式表（Stylesheet）语言的建议。但是在它的酝酿期，这条建议最终被分割成三个分立的标准。第一个，**XPath**，定义了一种在 XML 文档中定位信息的机制，并且它除了格式化（formatting）文档外还有许多其他的用途。第二个，**XSLT**，提供了一种将 XML 文档转化成其他数据格式，包括（但不仅限于）格式化语言的方法。最后，我们现在仅用术语 **XSL** 来命名一个在使用 XML 元素的文档中嵌入格式化信息的提议中的标准。

这三个标准同时也是相关的。它们一起提供了一种格式化 XML 文档的方法。XSLT 标准在很多情况下包括 XPath 的结构（construct），同时可以用 XSLT 将一个 XML 文档转化成 XSL 文档。它们既可以独立使用，也可以和其他技术一并使用。由于 XSL 格式化语言的发展不如 XSLT 成熟，并且也没有得到很好的支持，因此最初普遍认为 XSLT 主要用于从 XML 文档到可能用 **CSS** 风格化（styling）指令强化的 **HTML** 文档的转换。所以这里将深入阐述这两种格式。但是，这本书的前半部分侧重于把 XSLT 作为一个处理 XML 的一般化工具，并且将 XSLT 作为一种通过 XPath 找到并处理 XML 文档构件的方法。

目　　录

第三部分　引用

第 1 章　本书的使用

书的结构

本书第一部分，转换（XSLT），介绍了 XSLT 语言的特性。读者应该按照书的组织顺序阅读，因为每 1 章的内容都建立在以前各章所举实例的基础上。最起码也应该先阅读第 3 章和第 4 章。第二部分，格式化，介绍了 XSL、以及其他一些格式化语言和样式表（HTML 和 CSS）。第三部分，引用，包括了如何分析 XML 文档类型定义（DTD）和 XPath 标准的全部细节。

本书假定读者已经对 XML 的标记（markup）概念有所了解。读者至少应该比较熟悉它的语法、作用域（scope）、元素和属性的用途、内容和处理指令，而且对它的实体和 DTD 有初步的了解。

风格习惯

在 XSL、XSLT 以及相关技术中用粗体显示的名称和术语有明确的含义，它们通常在第一个重要出现位置以粗体显示，并在其后每一次进一步定义的地方都以粗体显示。

如下显示的正文 in a mono—spaced font 表示样本数据（example data），通常是一个 XSL 或者 XML 的片段。较大的样本从正文中分离：

```
This is a sample line of text.
```

尽管我们可能会使用粗体来强调一个样本的局部，比如‘look at **this** word’，但它并不具有如上所述的重要性。

当前材料的样本（打印或显示输出）显示如下：**This is presented material.** 斜体表示的单词或者是引用，或者是简单的希望引起注意。为清晰起见，元素和属性名在正文中大写，比如‘the Name element contains a name’，但是尽管 XML 标准规定名字是区分大小写的，在 XML 文档中元素和属性名都必须全部小写，例如‘<name>Smith</name>’。通常情况下 HTML 元素都以大写显示（但并不区分大小写）。

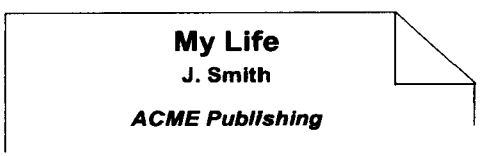
第2章 概 述

样式表

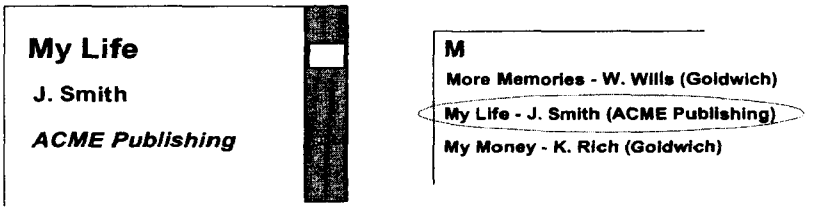
随着 1998 年 XML 的发布，广义标记（generalized markup）概念给我们带来的好处开始引起越来越多人的关注。如果使用正确，XML 标签能够标识出文档的每个部分；这样就能够定位、提取、分析并且重用这些文档片段。但是这种方法的一个缺点就是人们不能很好地显示这种使用了 XML 标签的文档。这些标签不能详细说明它所包含正文的显示格式。下面这个例子虽然详细地定义了书名、作者和出版商，但是并没有定义如何显示这些项目：

```
<book>
  <title>My Life</title>
  <author>J.  Smith</author>
  <publisher>ACME Publishing</publisher>
  ...
</book>
```

因此我们需要从其他地方获取样式表信息。通常我们从包含有规则集的数据文件中获取相应的样式表信息，规则集中的每个规则都详细定义了如何显示文档中的每个特定元素。我们就把这样的文件称作样式表。样式表已经产生许多年了，这期间它们形成了很多种形式，不同形式样式表的表现能力各不相同。至少我们可以用样式表来定义文本的字体和大小、文本显示风格（例如下划线、粗体或斜体显示），还可以定义段间距。特别地，有时还能定义前景色和背景色。大多数样式表都能自动给列表中的项目编号，增加例如‘警告：’的前置文本，甚至还能定义页面尺寸，定位页面的头、体和尾。更高级的样式表还能够对信息进行重排并且重复使用。使用样式表后，上述 XML 片段将以如下格式显示：



即使我们不需要使用样式表来定义 XML 文档的格式，样式表仍然是非常有用的。使用样式表的一个最重要的好处在于通过用一个样式表来代替另一个样式表，同样内容的文档就可以以不同方式显示。这样不同出版媒体就可以利用这一特性来吸引有不同要求的不同用户，或者创造出新的产品。



众所周知，尽管样式表不是显示 HTML 文档所必需的（这点不同于 XML 文档），但是 HTML 文档已经开始使用样式表了，这一事实充分证明了样式表是一个可行的概念。CSS（层叠式样式表，Cascading Style Sheets）就经常和 HTML 一起使用

大多数样式表都十分复杂，它们包括许多能够匹配源元素和输出格式的规则。即使非常简单的样式表也需要详细定义每条新规则从哪开始，同时标明每一个元素名和它所对应的输出方式。因此样式表必须符合一定的语法规范（specification），这就是样式表语言。广义标记概念有很长的历史，可以追溯到 XML 发布前很久，这一点可以从多种样式表语言已经存在了很多年得到证实。正式用于SGML(XML之父)样式表语言是DSSSL(Document Style and Semantics Specification Language），它是基于功能强大但非常难读的 SCHEME 计算机语言的。

本书使用了一种专用的样式表语言 FrameMaker+SGML，也就是一种基于 SGML 的样式表语言来定义它的格式。下列规则规定了这段的格式：

```
<Element>
   <Tag>Para</Tag>
   <Container>
      ...
      <TextFormatRules>
         <AllContextRules>
            <ParagraphFormatting>
               <PropertiesBasic>
                  <ParagraphSpacing>
                     <SpaceAbove>9pt</SpaceAbove>
                  </ParagraphSpacing>
                  <LineSpacing>
                     <Height>12.5pt</Height>
                  </LineSpacing>
                  </PropertiesBasic>
                  <PropertiesFont>
                     <Family>Times</Family>
                     <Size>10pt</Size>
                  </PropertiesFont>
                  ...
            </ParagraphFormatting>
         </AllContextRules>
      </TextFormatRules>
</Element>
```

XSL 标准

本书的标题提到了一个缩写词 XSL，它表示可延伸式样式表语言。在这种样式表语言发展的早期，我们使用这个名字来命名一种规范。从那时起，事情变得稍微复杂了一些，最终的规范实际上是三种标准 XSL、XSLT 和 XPath 的合并。这三种标准都有它们彼此独立的作用，同时它们一起能够完成对 XML 文档的格式化操作。

XSL

我们现在正以一种更狭义的方式使用最初的术语‘XSL’。XSL 标准定义了一种 XML 的应用，这种应用故意打破了 XML 文档结构的黄金规则（golden rule）。在这个应用中，元素用于描述内容的显示信息，而不是内容本身。在下面的例子中名称为‘block’的元素包含了一个文本块，比如一个自然段，而元素‘wrapper’则包含了要以特殊方式打扮的一部分文本：

```
<block>This is an <wrapper style="bold">XSL</wrapper>
document. </block>
```

这是 XSL 文件。

XML 文档是自描述的，因而就不应该以这种方式创建它。毕竟，使用了 XSL DTD 的 XML 文档对它内容的描述和 HTML 文档相类似，它实际上和 HTML 页非常像。和上例功能相同的 HTML 要稍微简单一些（尽管 HTML 是一种更受限制的格式）：

```
<P>This is an <B>HTML</B> document.</P>
```

这是 HTML 文件。

XSLT

XSLT（XSL 转换，**XSL Transformations**）是一种 XML 转换语言。使用 XSLT 很容易将 XML 文档转换成包括许多用于控制物体以指定方式表现的格式化语言在内的其他数据格式。这种语言最初作为 XSL 标准的一部分，现在却已经从 XSL 中分立出来形成它自己的标准了。XSLT 标准阐明了样式表文档的格式，同时包含了对映像规则（mapping rules）的定义。实际上 XSLT 文档就是符合 XSLT DTD 的 XML 文档。

当前我们主要用 XSLT 将 XML 文档转换成为可以被 Web 浏览器识别的数据格式。HTML 格式是和 XML 有着共同祖先的 SGML 的一种应用，因此也可以把 HTML 格式看作是一种 XML 格式（可能会出现一两个警告）。这样做会给我们提供很多便利，因为 XSLT 尤其擅长于将一个 XML 文档转换成另一个 XML 文档。下面这个例子将包含有强调术语的 XML 程序段，格式化成与之等价的 HTML 格式：

```
    <paragraph>A <emphasis>Highlighted</emphasis>
    term</paragraph>

<template match="paragraph">
  <html:P>
    <apply-templates/>
  </html:P>
</template>

<template match="emphasis">
  <html:B>
    <apply- templates/>
  </html:B>
</template>

    <P>A <B>Highlighted</B> term</P>
```

其实 XSLT 除用于格式化外还有许多其他用途。XML 文档很容易被转换成符合不同结构（可能符合不同 DTD）的 XML 文档。我们不仅可以重用、重新配置（reposition）、排列元素内容，从中分离或和其他内容进行合并，甚至可以将元素内容转换成属性值。

我们可以从 XSLT 样式表本身就是 XML 文档的事实中获益不少。首先，既可以用 DTD 来验证样式表的正确性，也可以利用 XML 解析器来指导样式表的作者。其次，我们可以将样式表存储在 XML 库中，要知道 XML 库可以对元素级内容进行操作和版本化。第三，由于 XSLT 样式表处理 XML 文档，这使我们能够将 XML 或者基于 SGML 的样式表转换成 XSLT 格式的样式表，并且根据另一个样式表创建出样式表变量。最后，样式表还可用来格式化用于显示或输出的其他样式表。

XPath

XSLT 可以将符合某个 DTD 的 XML 文档转换成符合不同 DTD 的另一个 XML 文档。最简单的做法是，用我们上面介绍的技术，把标签重新命名以使它们符合目标 DTD 的元素定义。比如说，元素‘Paragraph’既可以以元素‘Para’的形式输出，又可以以元素‘P’的形式输出。其实 XSLT 更强大的功能在于它允许内容的移动和复制。元素内容可以成为属性值，同时属性值也可成为元素内容。而且当元素出现在文档结构的重要位置时，我们可以对元素进行专门格式化。这种高级分析和对数据的操作是通过表达式语言（**expression language**）实现的。

最早的 XSL 提案包括这种表达式语言，但是现在的 XSLT 标准引用的是一个叫做 **XPath** 的分立标准。这条标准定义了一个查询 XML 文档结构的多用途语言。我们也可以把它作为一个查询语言，并用于高级超文本链接的架构。这种语言虽然不是基于 XML 的，但是它拥有能适用于不同上下文的紧凑语法。下面的表达式识别了包含在‘Chapter’元素并且‘Type’属性值为‘Important’的‘Paragraph’元素：

```
chapter//paragraph[@type='Important']
```

一旦我们理解了这个表达式语言，我们就可以用它来查询 XML 结构（通过使用 XQL）和链接结构（通过使用 Xpointer）。

XSL、XSLT 和 XPath 的合作

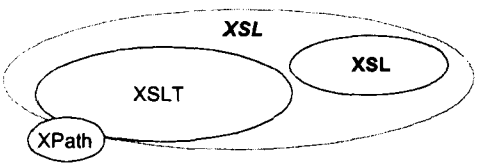
我们可以看到借助于 XPath 表达式，XSLT 提供了理想的、将任意 XML 文档转换成用于定义显示格式的 XSL 文档的方式。下面的例子给出了一条将源文档的‘Paragraph’元素映像成 XSL 格式下的粗体段落的 XSLT 规则，但是这种映像只有在段落包含在元素‘Chapter’内部并且‘Type’属性值为‘Important’时才正确，用 XPath 表示如下：

```
<template match="Chapter//Paragraph[@type='Important']">
  <fo:block font-style='bold'>
    <apply-templates/>
  </fo:block>
</template>
```

注意本书的重点在于讲述转换 XML 文档的 XSLT 标准，而此标准大量地使用了 XPath 表达式语言。实际上 XSL 标准本身在整本书中相对不是最重要的，因为一起使用 XSLT 和 XPath 除了创建 XSL 文档外还有许多其他用途。

命名冲突

我们现在用术语‘XSL’来描述三种不同的概念，其中的一个相对于其他两个更为精确一些。第一，‘XSL’正确命名了 XSL 标准本身。第二，虽然普遍但并不精确，它用来描述 XSLT 标准。第三，由于历史原因，我们所说的 XSL 通常既包含 XSLT（包括 XPath）又包含 XSL，其中用 XSLT 样式表来创建 XSL 文档。本书使用的是 XSL 第一和第三种定义，因为没有出现其他术语来描述（讨论中通过文章的上下文能够看出使用了哪种定义方式）。



尽管术语‘XSL 样式表’本应指输出 XSL 文档的 XSLT 文档，但我们通常将它理解为不管输出内容的任何 XSLT 文档。

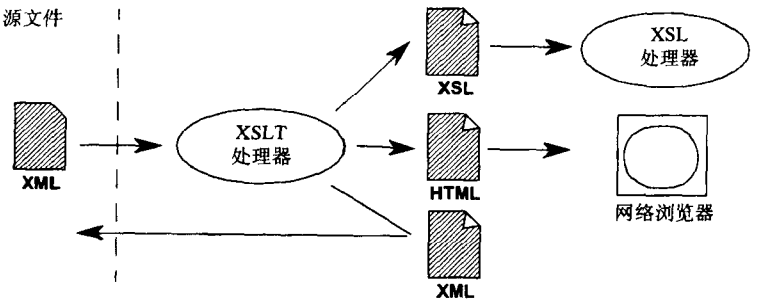
XSL/XSLT 处理器

现在已经出现了许多软件模块和应用来支持上述每一个标准。**XSLT 处理器**将任意的 XML 文档转换成为 XSL 文档，然后 **XSL 处理器**解释 XSL 格式化指令，并根据这些指令将

文档内容输出到纸张或屏幕上。

有些 XSLT/XSL 处理器会作为某种出版应用形成一个坚固的整体，例如网络浏览器或是台式出版软件包，这里处理器的输出被直接传到主应用程序输出。‘**嵌入式处理器**’的概念就可以用来讨论这类处理器。

但是有些 XSLT 处理器不会和表示应用程序紧紧地结合在一起，相反它们会产生一个应用程序可读的输出数据文件。在本书里，我们用术语‘**独立处理器**（standalone processor）’来描述这种处理器。独立处理器有多种不同的使用方式，它可以用来创建打扮成 XSL 的文档或者其他类型的格式化文件（比如 HTML），或者用来创建一个新的 XML 文档。



为什么需要 XSL

我们身边已经存在许多样式表语言了，在这种情况下一种新样式表语言的产生一定有它充分的原因。实际上有些人也提出 XSL 并不是必要的，但是这种语言的发展的确有它的理由。

首先，XSL 和其他大量专门产品的样式表语言相比有明显的优势。作为一个工业标准，它一开始就被大量的工具支持，并且配置在大范围的计算机操作平台上。许多浏览器和出版应用程序都在努力实现能够接受打扮成 XSL 的表示文档，这样就脱离了对特别供应商的依赖性。当然这些便利都是对使用 XML 本身的一种补充。但是现在仍然有少量现有标准值得我们讨论。

当前 CSS（层叠式样式表语言）格式的广泛使用部分是由于主流网络浏览器厂商已经在他们的产品中加进了对这种语言的支持。最初开发 CSS 时明确地把它作为一种对 HTML 的补充。刚开始听到这种说法的时候感到很奇怪，因为 HTML 本身包含大量专门格式化的标签，所以这样的一个样式表实际上并不是必要的。实际上最早提出 CSS 是为了给网页作者提供对网页格式的更多控制，它为我们提供了一种重新设置浏览器默认格式设置的简单方法，并且在它最简单的形式下，格式化指令甚至不需存储在一个单独的样式表中，而只需要把它存储在一个称作‘Style’的 HTML 属性中。CSS 可以和 XML 一起使用，有些人