

目 录

第一部：XSLT 学习之旅

第 1 章 简介.....	1
1.1 XSL 和 XSLT.....	4
1.1.1 XSL 的处理流程.....	5
1.2 XSLT 的用途.....	6
1.2.1 XSLT 的处理流程.....	6
1.2.2 网站应用.....	7
1.2.3 通信应用.....	8
1.3 XSLT 和 XPath.....	9
1.4 XSLT 处理器.....	9
1.4.1 Apache Xalan.....	9
1.4.2 Microsoft MSXML.....	10
1.5 简单范例.....	10
1.5.1 XML 输入文件.....	10
1.5.2 XSL 样式表文件.....	11
1.5.3 XML 输出文件.....	14
1.6 统一字符编码.....	16
1.6.1 统一字符编码写 XML 文件.....	17
1.6.2 用字参码改写 cbook.xml.....	17
第 2 章 模板结构.....	21
2.1 模板结构解析.....	22
2.2 XSLT 顶层组件.....	24
2-2-1 XML 组件和属性名称.....	28
2.3 模板模块.....	30
第 3 章 模板规则.....	35
3.1 结构树.....	36
3.2 XSL 模板规则.....	42
3.2.1 内建模板规则.....	44
3.2.2 xsl:apply-templates 组件.....	48
3.2.3 xsl:value-of 组件.....	51
3.2.4 personal.html.xsl.....	54

3.2.5	songpoetry.html.xsl.....	65
3.3	xml-stylesheet 处理命令	66
第 4 章	设计模板规则.....	69
4.1	对比样式	70
4.1.1	用/对比子组件	70
4.1.2	用//对比子孙组件	76
4.1.3	用@对比属性.....	76
4.1.4	空格和 xml:space	81
4.1.5	使用 运算符.....	88
4.1.6	[]的测试条件	89
4.1.7	用 id()对比组件	91
4.1.8	用 comment()对比注释	96
4.1.9	用 processing-instruction()对比处理命令	97
4.1.10	用 text()对比文字结点	97
4.2	XPath 表达式.....	98
4.2.1	位置路径	99
4.2.2	简写语法	106
4.2.3	XPath 核心函数库	108
4.2.4	表达式种类	111
4.2.5	属性值模板	117
4.3	XSLT 命令.....	121
4.3.1	xsl:for-each 命令	121
4.3.2	xsl:copy 命令	122
4.3.3	xsl:copy-of 命令	123
4.3.4	xsl:sort 命令	125
4.3.5	xsl:variable 命令	130
4.3.6	xsl:param 指令.....	133
4.3.7	xsl:call-template 指令	134
4.3.8	xsl:number 指令	136
4.3.9	xsl:attribute-set 指令	141
4.3.10	xsl:key 指令	142
4.3.11	xsl:message 指令	144
4.3.12	xsl:namespace-alias 指令	146
4.3.13	xsl:fallback 指令	149
4.4	XSLT 函数.....	159
4.4.1	document()函数	159
4.4.2	current()函数	162
4.4.3	unparsed-entity-uri()函数	162

4.4.4	generate-id()函数.....	164
4.4.5	system-property()函数.....	164
4.5	XHTML	165
第 5 章	XSLT 处理器.....	173
5.1	Xalan-Java 处理器.....	174
5.1.1	Xalan-Java 处理器的特点.....	174
5.1.2	执行 Xalan-Java 处理器.....	174
5.1.3	Java 应用程序	176
5.1.4	Java 服务件	184
5.2	Xalan-C++处理器.....	205
5.2.1	Xalan-C++处理器的特色.....	205
5.2.2	执行 Xalan-C++处理器	206
5.2.3	C++应用程序	207
5.2.4	ApacheModuleXSLT 模块.....	214
第 6 章	网页排版系统.....	217
6.1	安装 Cocoon 排版系统	218
6.2	测试	222
6.3	网页排版系统服务.....	223

第二部：XSLT 参考文献

第 7 章	XSLT 总论	233
7.1	XSLT 处理模式.....	234
7.1.1	结构树：数据模型	235
7.1.2	来源树之前	240
7.1.3	输出格式	243
7.1.4	转换机制	244
7.1.5	控制结点处理路线	245
7.2	数据类型	245
7.3	变量和参数.....	247
7.4	表达式	247
7.5	模板结构	248
7.5.1	模块化结构	248
7.5.2	xsl:stylesheet 组件.....	249
7.5.3	顶层组件	249
7.5.4	模板	250

7.6	转换时机	254
第 8 章	XSLT 组件	255
8.1	模板规则	257
8.1.1	xsl:template	257
8.1.2	xsl:apply-templates	272
8.1.3	xsl:call-template	282
8.2	模板结构	291
8.2.1	xsl:stylesheet	292
8.2.2	xsl:transform	305
8.2.3	xsl:import	305
8.2.4	xsl:include	312
8.2.5	xsl:apply-imports	317
8.3	输出	320
8.3.1	xsl:value-of	320
8.3.2	xsl:element	323
8.3.3	xsl:attribute	327
8.3.4	xsl:comment	331
8.3.5	xsl:processing-instruction	332
8.3.6	xsl:text	334
8.3.7	xsl:attribute-set	337
8.4	变量和参数	348
8.4.1	xsl:variable	348
8.4.2	xsl:param	354
8.4.3	xsl:with-param	364
8.5	复制	366
8.5.1	xsl:copy	366
8.5.2	xsl:copy-of	370
8.6	条件式处理	372
8.6.1	xsl:if	372
8.6.2	xsl:choose	376
8.6.3	xsl:when	378
8.6.4	xsl:otherwise	380
8.7	重复	384
8.7.1	xsl:for-each	384
8.8	排序和编号	387
8.8.1	xsl:sort	387
8.8.2	xsl:number	394
8.8.3	xsl:decimal-format	420

8.9	搜索	423
8.9.1	xsl:key	423
8.10	控制空格结点	431
8.10.1	xsl:strip-space	431
8.10.2	xsl:preserve-space	434
8.11	输出格式	437
8.11.1	xsl:output	437
8.12	其他	449
8.12.1	xsl:fallback	449
8.12.2	xsl:message	455
8.12.3	xsl:namespace-alias	458
第 9 章	XPath 表达式	463
9-1	位置路径	465
9.1.1	位置路径	467
9.1.2	取向轴	468
9.1.3	结点测试	469
9.1.4	谓词	471
9.1.5	简写语法	471
9.2	表达式	473
9.2.1	基本表达式	473
9.2.2	函数调用表达式	473
9.2.3	结点集表达式	474
9.2.4	布尔表达式	474
9.2.5	数值表达式	476
9.2.6	字符串	477
9.2.7	语汇结构	477
9.3	EBNF	478
9.4	XSL 模板使用 XPath 表达式	480
第 10 章	XSLT 对比样式	481
10.1	制程规则	482
第 11 章	函数	485
11.1	结点集函数	487
11.1.1	last()	487
11.1.2	position()	492
11.1.3	count()	497
11.1.4	id()	500

11.1.5	local-name()	502
11.1.6	namespace-uri()	505
11.1.7	name()	506
11.2	字符串函数	510
11.2.1	string()	510
11.2.2	concat()	512
11.2.3	starts-with()	514
11.2.4	contains()	515
11.2.5	substring-before()	517
11.2.6	substring-after()	520
11.2.7	substring()	524
11.2.8	string-length()	526
11.2.9	normalize-space()	528
11.2.10	translate()	531
11.3	布尔函数	536
11.3.1	boolean()	536
11.3.2	not()	537
11.3.3	true()	539
11.3.4	false()	539
11.3.5	lang()	540
11.4	数值函数	544
11.4.1	number()	544
11.4.2	sum()	545
11.4.3	floor()	548
11.4.4	ceiling()	549
11.4.5	round()	550
11.5	XSLT 函数	552
11.5.1	document()	552
11.5.2	key()	560
11.5.3	format-number()	562
11.5.4	current()	564
11.5.5	unparsed-entity-uri()	569
11.5.6	generate-id()	570
11.5.7	system-property()	572
11.5.8	element-available()	574
11.5.9	function-available()	575

第 12 章 通用统一字符编码 577

12.1	何谓统一字符编码	578
------	----------	-----

目录

12.1.1	ASCII 码与统一字符编码	578
12.1.2	字节与字符	579
12.2	统一字符编码字符集的结构	580
12.3	统一字符编码字符数据库	582
12.4	统一字符编码标准规范与 ISO/IEC 10646 国际标准规范	583
12.4.1	ISO/IEC 10646 国际标准规范的结构	584
12.5	编码	585
12.5.1	UTF-16	586
12.5.2	UTF-8	587
12.5.3	字节次序	588
12.6	转码	589
附录 A	XSLT 产品	591
附录 B	网站资源	595
附录 C	中英文名词对照表	599
附录 D	明日 XSLT	611
附录 E	参考文献	613



第一部着重探讨 XSLT 转换语言各种构件的重点用法（含 XPath），以及 XSLT 转换在互联网上的实务应用。核心目标是希望读者读完第一部之后，就有能力自行设计和撰写 XSL 模板文件，从而领会 XSLT 这种新兴技术的实用潜力。

第一部的各章各节并没有详列各种构件的相关细节，这就是为什么这一部的标题是“XSLT 学习之旅”的原因。偶尔谈到一些难以一笔带过的细节时，会出现比较严谨的论述，这是为了避免初学者误解其意而采取的必然措施；读者宜耐心研读。

第一部共计 6 章，章名次序如此：第 1 章<<简介>>，第 2 章<<模板结构>>，第 3 章<<模板规则>>，第 4 章<<设计模板规则>>，第 5 章<<XSLT 处理器>>，第 6 章<<网页排版系统>>。

XSLT

学习之旅

第一部

第1章

简介



XML 指的是可扩展标记语言 (EXtensible Markup Language), 运用一组组成对的标记 (Tag) 按其从属关系组织数据的结构 (Structure), 而且拥有简洁的语法 (Syntax), 是目前电子商务数据交换的主流语言。然而, 单凭 XML 却是寸步难行、无以成事, 这是因为 XML 只定义数据的结构和语法规则, 至于数据本身带有何种语义, 或者说是语法 (Grammar), 则由各类应用软件根据不同 XML 文件 (Document) 的内容 (Content) 予以解读和处理。

到目前为止, 如果你想应用软件处理 XML 数据, 一定会碰到 SAX (Simple API for XML) 和 DOM (Document Object Model) 这两种低级界面。无论是哪一种, 你必须先了解 SAX 或 DOM 组织数据的原理和过程, 才能利用程序语言提供的 SAX 或 DOM 的标准 API 来操控 XML 文件。

所幸, XSLT 定义了一套转换语言 (Transformation Language), 把操作 XML 数据的层次推向高级语言的境界。概括而论, 只要声明一组一组符合 XSLT 语义和语法要求的规则, 再交给 XSLT 处理器 (XSLT Processor) 解译, 就能对 XML 数据做各种处理, 诸如搜索、排序、统计等, 进而得到新的信息, 以供后续之用。例如, 服务器获取 XSLT 处理器的输出之后, 将结果返回客户端, 显示在浏览器让用户观看。

虽然 XSLT 是处理 XML 数据的高级语言, 毕竟还是伴随 XML 而生的新兴技术之一, 探索空间非常的大, 而且和 XML 其他家族技术集成之后 (例如, XPath), 多少引进一些难懂的新概念, 是初学者在应用 XSLT 时要花一点时间克服的两个难题。

1.1 XSL 和 XSLT

前面提到 XSLT 是伴随 XML 而生的新兴技术, 其实并不完全正确, 因为 XSLT 是附属 XSL 的转换语言, 最初的设计目的全是为了配合 XSL, 所以, 比较贴切的说法应该是: XSLT 是因 XSL 才得以出现的新兴技术。

XSL 指的是可扩展样式表语言 (eXtensible Stylesheet Language), 也就是用来写 XSL 样式表文件的语言。XSL 样式表文件和 XML 有密切的关系。一般而言, XML 标记只用来组织数据间的关系, 表达数据的结构, 不像有些 HTML 标记除了存储数据之外, 还会携带设置格式的信息, 例如, <H1>这是 H1</H1>。至于 XML 数据最后要用什么形式展现出来, 则交给样式表文件决定, 可能是 PDF、PostScript、HTML 或 WML。

把数据和页面设置 (Aresentation) 分开, 使相同的数据可以用不同的形式显示出来, 这样的设计架构具有许多的优点。首先, 由于数据本身不夹带排版信息, 数据就能保有纯粹性, 可供移植和再用。另外, 阅读数据的人也有机会照自己的喜好决定要用哪一种页面设置来观看文件内容。网页设计师在设计网页内容时, 也可以暂时抛开页面设置的束缚, 专心内容的设计, 在网页内容备妥时, 再套上样式表定出版型即可。这就是 XSL 在 XML 世界中扮演的角色。

当 XSL 样式表文件写好时, XSL 样式表处理器 (XSL Stylesheet Processor) 就可以决定 XML 文件的数据要用什么格式展现出来。因此, 同一份 XML 文件可以拥有好几份样式表文件, 依需求决定要以何种面貌显示出来。

1.1.1 XSL 的处理流程

XSL 样式表处理器解译 XSL 样式表文件的过程在概念上可分为两个步骤：

✚ 第一阶段：结构树转换（Tree Transformation）

✚ 第二阶段：设置格式（Formatting）

第一阶段在转换结构树时，XSL 样式表处理器会用到 XSL 转换器（Transformer），而第二阶段在设置格式时，XSL 样式表处理器会用到 XSL 格式器（Formatter）。在结构树转换阶段，XSL 转换器会建造三棵结构树（Tree）：一是来自 XML 文件的来源树（Source Tree），二是来自 XSL 样式表文件的样式表树（Stylesheet Tree），三是从来源树和样式表树得到的结果树（Result Tree）。得到结果树之后，就进入设置格式的阶段。XSL 格式器会解读结果树，根据结果树中的格式对象（Formatting Object, FO）决定数据的排版格式，最后得出结果，诸如 PDF 文件（.pdf），或 PostScript 文件（.ps）。XSL 的格式对象统称为格式语言（Formatting Language）。格式对象最初是放在 XSL 样式表文件中，通过 XSL 转换器的转换，将格式对象置入结果树中，XSL 格式器再解读格式对象携带的信息做出该有的排版效果。我们可以用图 1.1 的流程图来说明这两个阶段。

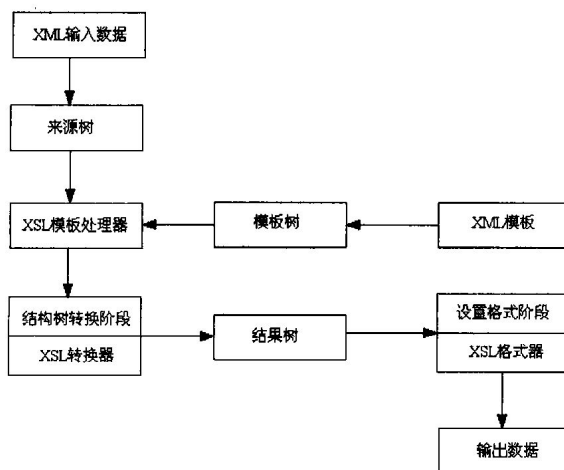


图 1.1 XSL 样式表处理器的处理流程

这是 XSL 样式表处理器标准的分解过程，就实务而言，这两个阶段不必分得如此清楚，也可以合成一个步骤，在转换的同时就把该有的格式设计好。另外，这两个阶段也可以拆开来分别进行，也就是 XSL 转换器和 XSL 格式器彼此独立毫不相干。换句话说，XSL 样式表文件中不见得要放格式对象，这是因为结果树不见得会交给 XSL 格式器设计排版。由于这两个阶段的工作可以分开来设计，而且目前设置格式阶段的规范并未臻于完善，因此，很多 XSL 工具都只有针对第一阶段的结构树转换，设计出符合规范要求的 XSL 转换器。

XSLT 出场的时刻就是第一阶段的结构树转换。XSLT 指的是可扩展样式表转换语言（EXtensible StyleSheet Language: Transformation），简称 XSL（T）转换语言，目的就是让 XSL 转换器把来源树按照样式表树的意志转换成结果树；转换的过程完全是结构树对结构树的关系。

由上述讨论可知，XSL 可分为 XSL 转换语言和 XSL 格式语言两种，分别在结构树转换阶段和格式设置阶段起关键性的作用。虽然 XSL 样式表文件中可以同时含有 XSL 转换语言和 XSL 格式语言的组件（element），然而这本书探讨的是 XSLT，将 XSLT 视为独立的技术，因此，书中不会触及 FO 的部分。

一般而言，只针对结构树转换阶段设计的 XSL 转换器就另称为 XSLT 处理器，而 XSLT 处理器使用的模板则仍称为 XSL 模板，然而，由于其目的仅止于结构的转换，因此，称为 XSLT 转换模板或 XSLT 模板也未尝不可。

此外，图 1.1 展现出来的 XSL 处理流程中最重要的共同性质就是所有牵涉在其中的文件都是 XML 文件。输入数据是 XML 文件，XSL 模板也是 XML 文件，而输出的数据也是 XML 文件。因此，无论在哪个阶段，你接触的都是 XML，这样便形成了自给自足的系统。在编辑 XML 时，只需用同一套编辑工具即可；在运用 XML 时，也只需 XML 的知识。

1.2 XSLT 的用途

上一节是让读者了解 XSLT 在 XSL 排版设计中所扮演的角色。如果 XSLT 要和 XSL 搭配使用，则格式对象和 XSL 转换语言一定会出现在 XSL 样式表文件中。但是，如果 XSLT 要独立使用，XSL 样式表文件中只需有 XSL 转换语言，即可达到转换的目的。所以，到了这里，读者可以暂时忘记 XSL 格式语言了。也许你会问，既然不考虑排版的问题，单凭 XSLT 又能转换出什么？要探讨这个问题，就让我们先从 XSLT 的定义着手。

1.2.1 XSLT 的处理流程

根据 XSLT 1.0 规范的定义，XSLT 是一种转换语言，可以把 XML 文件转换成别的 XML 文件。XSLT 处理器的处理流程可用图 1.2 来说明：

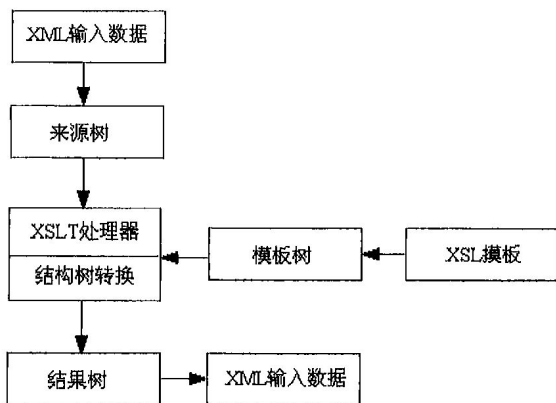


图 1.2 XSLT 处理器的处理流程

XSLT 规范的定义是从表象来说的，其实从一种 XML 数据转换成另一种 XML 数据的过程中，无论输入还是输出，都是 XML 数据。然而，两者间最足以道称的差别在于转换前和转换后的数据结构可以相差非常的大。把握这个核心概念，XSLT 真正的目的其实就是用于

转换 XML 数据的结构。

1.2.2 网站应用

当你在写 XML 文件时，第一件要考虑的事就是数据结构：一旦决定哪个组件要放在哪个组件之后，XML 文件的数据结构就定型了。如果你懂一点数据库，大概会听过数据模型（Data Model）这个术语。设计数据库时，第一步就是决定数据模型。XML 文件既然是存储数据的场所，其实和数据库的性质十分相似，差别在于 XML 是用纯文本标记间的层次和主从关系来设计其数据模型。传统的数据库有标准 SQL 语言可供操控设计好的数据库；而如今的 XML 文件则有标准 XSLT 转换语言可与之匹敌。没有错，如果和关系数据库（Relational Database）相比拟，XSLT 的确可称之为 XML 的 SQL 语言。由此可知，当初 XSL 规范的设计团队在设计 XSLT 时，能够考虑到 XSLT 和 XSL 脱勾时独立应用的价值，确实是让 XSLT 得以放开手脚的明智之举。

既然 XSLT 可作为 XML 的 SQL 语言，很自然的你会联想到 XSLT 在网站上的应用。XML 是当今电子商务的标准转换语言，后端数据库如果能改采 XML 作为存储格式，将有利于企业系统的集成。但是，一般预测，XML 要完全替代 HTML 在前端网页中的地位，恐怕还要十来年的时间。况且，XHTML（EXtensible HyperText Markup Language）使得 HTML 在 XML 世界中重生，我们甚至可以大胆地说，HTML 在可预见的将来是不会消失的，HTML 只是换个面貌，以比较严谨的 XML 语法改写 HTML 继续在网页文件设计中扮演主要角色。因此，如何把 XML 后端数据库的数据转换成 HTML 或 XHTML 网页文件，就是一项重要的实务课题。

XSLT 就是这个转换过程的黑箱（Black Box）。图 1.3 说明了这个转换过程。

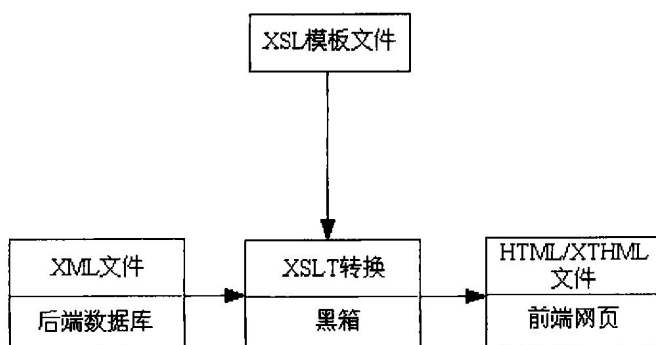


图 1.3 XML 转 HTML/XHTML

虽然 XSLT 的基本定义是把 XML 文件转换成另一种 XML 文件。但是，HTML 实在是太流行了，因此，XSLT 也可以转换出 HTML 文件。在图 1.3 这样的架构下，前端网页最后的解释权是由网页浏览器负责解读，浏览器会根据文件中采用的 HTML 标记和存储的数据决定其在浏览器窗口中的排版格式。由此可知，XSL 格式器其实并未消失。网页浏览器中的版面设计引擎（rendering engine）就是 XSLT 独立运作时来自外部世界的 XSL 格式器，这正是结构树转换阶段和设置格式阶段分开处理的典型范例。另外，XHTML 文件本身就是 XML，所

以没有意义。

目前主流的网页浏览器都支持 XML，诸如 MSIE 5/6 和 NN 6/Mozilla 5。也就是说网站服务器 (Web Server) 可以直接供应 XML 文件和 CSS 样式表文件，当客户端的浏览器请求 XML 文件并连同 CSS 样式表文件一并下载之后，就可以根据随附之 CSS 样式表文件决定 XML 文件在浏览器窗口中的排版格式。由此可知，XSLT 在这又不会缺席了。

另外，提供 WAP 手机上网是网站逐渐新增的功能，虽然不见得人人都会一机在手，也不见得人人都想使用 WAP 手机上网，但是，网站能够多提供一点附加价值的服务，总不会是坏事一件。WAP 手机用的网页是 WML 网页，而 WML 指的是无线标记语言 (Wireless Markup Language)。XSLT 也可以把 XML 文件转换成 WAP 手机所需的 WML 网页。

最后，XSLT 也可以把 XML 文件转换成纯文本的文件，也就是一般所说的纯文本文件 (Plain Text File)。换句话说，你只要准备好 XML 文件和各种 XSL 样式表文件，任何人就可以按其需求转换出所需的格式，包括纯文本文件。事实上，转换成纯文本文件有一个很实在的用途，如果你发现设计出来的 XML 文件必须大量修改数据结构，说不定重建新数据结构会比较有效率，这个时候，就可以用 XSLT 把原 XML 文件转成纯文本文件，直接消去所有的标记，再另图新的数据模型。

由上述讨论可知，归根究底，就是呼应前面所提的深层定义：XSLT 是转换 XML 数据结构的转换语言。相信你会发现一项事实，XSL 样式表文件是 XSLT 转换过程中不可缺少的要件，因为 XSLT 转换语言的组件就是放在 XSL 样式表文件中，再交给 XSLT 处理器解读语法和语义将 XML 来源文件转换成最终的输出文件。既然如此，学习 XSLT 其实就是在学 XSL 样式表文件的写法，以及 XSLT 转换语言众多构件的用法。

1.2.3 通信应用

除了网站方面的应用之外，XSLT 在任何 XML 应用软件间的通信也有不可忽视的发展潜力。具体言之，转换 XML 数据结构就是指重新组织 XML 数据，由于数据能重新组织，XML 应用软件才得以免受数据格式限制之苦。XML 应用软件间如需传输数据，只要通过 XSLT 转换，即能达到传输目的，而不用为了将数据存成其他 XML 应用软件所需的格式而花费精力编写该 XML 应用软件专用的数据转换模块。图 1.4 即为这个概念的图解说明。

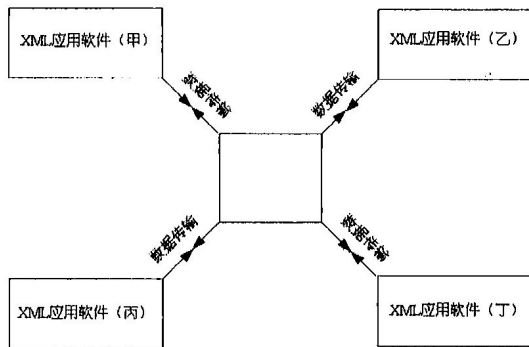


图 1.4 不同 XML 应用软件间的数据传输转换

1.3 XSLT 和 XPath

XPath 指的是 XML 路径语言 (XML Path Language)，主要目的是定出 XML 文件中各个部位的路径位置。XSLT 处理器在处理结构树时，必须利用 XPath 路径语言才能在结构树中来回游走。第 3 章讨论结构树的数据模型时，会开始遇上 XPath。这本书读通之后，你自然能看清楚 XSLT 和 XPath 之间的关系。

1.4 XSLT 处理器

XSLT 处理器的目的就是依 XSL 样式表文件定义的规则，把 XML 文件实际转成另一种 XML 文件。目前 XSLT 处理器的产品种类众多，本节介绍其中两种，足以应付 XSLT 的应用。其他产品的信息可参考附录 A。最新的信息可上网查询，例如，W3C 协会的主网站就是一个不错的起点：

<http://www.w3c.org/>

这一节要介绍的两种 XSLT 处理器分别是 Apache 的 Xalan 以及 Microsoft 的 MSXML，其网址如下所示：

✚ Apache Xalan 的网址：<http://xml.apache.org/>。

✚ Microsoft MSXML 的网址：<http://msdn.microsoft.com/xml/>。

目前来讲，XSLT 处理器的工具语言 (Implementation Language) 主要有 C++、Java 这两种主流的面向对象语言 (Object-Oriented Language)，其中以 Java 的产品最丰富，因此，你要先把相关的程序语言软件，或者是工具库准备好，安装 XSLT 处理器才有意义。

应用软件可以免费下载，通常就表示你必须自行 DIY 的地方非常的多，而且产品多半不带任何担保，无法顺利安装或正常使用的情況难保不会发生在你身上，因此，读者应尝试安装几种 XSLT 处理器（从这一节介绍的和附录 A 列出来的列表中选几样），试用一段时间，再决定要如何取舍。虽然是免费下载，并不表示毫无版权，你应当遵照每种产品的使用说明制定的各种条款。

有关 XSLT 处理器的安装，请参考阿熏的网站：

<http://kaoru.zzz.com.tw/>

<http://kaoru.zzz.com.tw/index.html>

以下扼要介绍这两种 XSLT 处理器。

1.4.1 Apache Xalan

Apache 是相当知名的网站服务器，主网站是 <http://www.apache.org/>，其配属的相关计划十分庞杂，而我们的焦点是其中的 Apache XML 计划。Apache XML 计划的目标之一是提供企业级的 XML 实务方案，然而研发过程却是采取开放合作的模式。

Apache XML 下有许多子计划，目前有 Xerces、Xalan、Cocoon、FOP、Xang、SOAP、Batik 和 Crimson 等。各子计划的详情请参照网页上的信息，我们关心的是其中的 XSLT 处理器：Xalan。

Xalan 完整实作了 W3C 制订的 XSLT 1.0 和 XPath 1.0 规范, 目前有 Java 和 C++ 这两种语言的实作版本, 两者都可从命令行执行, 而且 Java 版的 Xalan 还可以用在应用件 (applet)、服务件 (servlet), 或其他各种 Java 应用程序的模块中。C++ 版的 Xalan 当然在性能上获取领先的地位, 而且同样可以用在各种 C++ 应用程序的模块中。

1.4.2 Microsoft MSXML

MSXML 是 Microsoft 提供的 XML 工具库, 分成 MSXML Parser 和 MSXML SDK 两部分。MSXML Parser 支持 XSLT、XPath、SAX2 和 DOM 等, 而 MSXML SDK 则含有标头文件 (Header File)、.lib 文件以及 MSXML 的说明文件 (XSLT、XPath 和 SAX2 等)。MSXML SDK 安装之后, 文件会放在 \Program Files\Microsoft XML Parser SDK\文件夹内, 有三个子文件夹, 其中的 Docs\放的就是 MSXML 的说明文件 xmlsdk30, 用鼠标双击, 可以看到如图 1.5 的窗口画面。

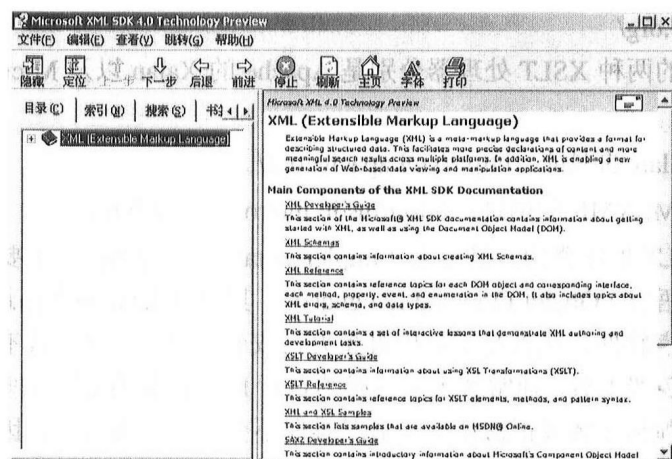


图 1.5

Microsoft 提供的 XSLT 命令行工具是 MSXSL (Msxsl.exe)。MSXSL 只是一个小程序, 当你从命令行执行时, MSXSL 用的还是 MSXML 的 DLL 工具库。

1.5 简单范例

前面谈了这么多东西, 如果没有实际做过 XSLT 转换, 恐怕读者心中还是会打个问号。照前面所说的, XSLT 转换的要件有三: XML 文件, XSL 样式表, 以及合乎 XSLT 规范要求的 XSLT 处理器。这里举的范例只是要让读者用上一节建议的 XSLT 处理器亲自做一遍 XSLT 转换, 所以你不用细看 XSL 样式表文件的内容, 本书后面的章节自会逐一说明。

1.5.1 XML 输入文件

XML 输入文件列于范例 1.1, 名为 book.xml