

附录E IE 5 XSL引用

IE 5广泛支持W3C在1998年12月16日发布的XSL草案的转换（ Transformations ）部分，但是它们之间也存在着一些微小的差异。 IE 5不支持格式化对象（ Formatting Objects ）和流对象（ Flow Objects ）建议。本附录详细描述了IE 5的最终版本对XSL的支持。

XSL定义了一组在 xsl命名空间有特殊含义的XML元素（即：每个元素都以 xsl命名空间标识符为前缀）。这些元素能够将文档转换为一种新的格式。在 W3C建议中，格式化对象用于定义每种元素转换的真正输出格式。在IE5中，我们通常使用HTML定义转换的新文档格式。

另外，XSL还能够将任何XML文档转换为其他（不同的）XML文档，或者转换为采用其他格式的文档。例如，它能够将一个 XSL样式表文档转换为另一个 XSL样式表文档，或者转换为专为其应用程序定制的格式。

E.1 IE 5 XSL元素

IE 5中的XSL提供了二十种元素，它们用于创建 XSL样式表或XML文档中的样式部分。下表列出了这些元素。

表 E-1

名 称	描 述
xsl:apply-templates	在模板中使用，用于指示XSL寻找另一个特定的模板，并应用于本节点。它包含以下属性： order-by="["+ "-" xsl-pattern"]" select="xsl-pattern"
xsl:attribute	用于创建新的 Attribute节点，并将它附加在当前的元素上。它仅包含一个属性： name="attribute-name"
xsl:cdata	用于在输出的当前位置创建新的 CDATASection。它没有属性
xsl:choose	与xsl:when和 xsl:otherwise配合使用，根据相同或不同节点的某个条件提供选择机制。类似于 If...ElseIf...Else结构。它没有属性
xsl:comment	用于在输出的当前位置创建新的 Comment节点。它没有属性
xsl:copy	用于将当前节点完整地拷贝到输出。它没有属性。
xsl:define-template-set	用于定义一组在样式表中有特殊范围的模板。它没有属性
xsl:element	用于在输出的当前位置创建新的 Element节点。它仅包含一个属性： name="element-name"
xsl:entity-ref	用于在输出的当前位置创建新的 EntityReference节点。它仅包含一个属性： name="entity-reference-name"

(续)

名 称	描 述
xsl:eval	用于计算字符串表达式，并将结果插入输出。其中字符串可以是数学表达式、逻辑表达式、XSL函数或定制的脚本函数。它仅包含一个属性： language="language-name"
xsl:for-each	用于创建类似于 For...Next 循环的循环结构，使得同一模板可以应用于多个节点。它包含以下属性： order-by="[+ -] xsl-pattern" select="xsl-pattern"
xsl:if	用于在模板中创建条件分支，与 If...Then 结构类似，使得一个模板能够根据条件提供不同的输出。它仅包含一个属性： match="condition-pattern"
xsl:node-name	用于将当前节点的名称以文本串的形式插入输出。它没有属性
xsl:otherwise	参见 xsl:choose。它没有属性
xsl:pi	用于在输出的当前位置创建新的 ProcessingInstruction 节点。它仅包含一个属性： name="processing-instruction-name"
xsl:script	用于定义包含全局变量声明和脚本代码函数的模板区域。它仅包含一个属性： language="language-name"
xsl:stylesheet	用于定义 XSL 样式表的“根”元素，所用的脚本语言，创建输出文档时是否保留输入文档中的所有空白，以及 xsl 前缀的命名空间声明 它包含以下属性： xmlns:xsl="http://www.w3.org/TR/WD-xsl" language="language-name" indent-result="[yes no]"（缺省值为“no”） 注意：在 IE5 的 XSL 中，此处必须出现命名空间
xsl:template	用于定义模板，它所包含的指令用于将 XML 输入中匹配特定模式的节点转换到输出中。它包含以下属性： language="language-name" match="xsl-pattern"
xsl:value-of	用于计算 select 属性中的 XSL 模式，并将匹配的节点及其子孙节点的值以文本形式插入模板。它仅包含一个属性： select="xsl-pattern"
xsl:when	参见 xsl:choose。它仅包含一个属性： match="xsl-pattern"

E.1.1 XSL 样式表的结构

下面的例子说明了如何利用 XSL 元素构造 XSL 样式表，同时显示了 XSL 元素能够创建的结构类型。由于大多数元素都能够互相嵌套，因此下面的例子仅仅展示了一小部分可能出现的组合。然而，一般而言，每个样式表都包含一个与文档的根元素匹配的模板，以及若干应用于文档中特定元素的特殊样式。

程序清单 E-1

```

<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

  <xsl:template match="...">
    <xsl:value-of select="..." />
    <xsl:eval> ... </xsl:eval>
    <xsl:if match="..."> ... </xsl:if>
    <xsl:copy />

    <xsl:choose>
      <xsl:when match="..."> ... </xsl:when>
      <xsl:otherwise> ... </xsl:otherwise>
    </xsl:choose>

    <xsl:for-each select="...">
      <xsl:value-of select="..." />
      <xsl:eval> ... </xsl:eval>
      <xsl:if match="..."> ... </xsl:if>
      <xsl:copy />
      <xsl:apply-templates />
    </xsl:for-each>

    <xsl:apply-templates select="..." />
  </xsl:template>

  <xsl:define-template-set>
    <xsl:template match="..."> ... </xsl:template>
    <xsl:template match="..."> ... </xsl:template>
  </xsl:define-template-set>

  <xsl:script> ... </xsl:script>

</xsl:stylesheet>

```

在XSL中创建新节点

以下XSL元素都能够在输出文档中创建新节点：xsl:attribute，xsl:cdata，xsl:comment，xsl:element，xsl:entity-ref和xsl:pi。

以下代码用于创建XML节点<![CDATA[This is a CDATA section]]>：

```
<xsl:cdata>This is a CDATA section</xsl:cdata>
```

以下代码用于创建XML节点<!ENTITY copy "©">：

```
<xsl:entity-ref name="copy">©</entity-ref>
```

以下代码用于创建XML节点<!--This is the comment text-->：

```
<xsl:comment>This is the comment text</xsl:comment>
```

以下代码用于创建XML节点<?WroxFormat="StartParagraph"?>：

```
<xsl:pi name="WroxFormat">StartParagraph</xsl:pi>
```

以下代码用于创建XML元素<title>Instant JavaScript</title>：

```
<xsl:element name="title">Instant JavaScript</xsl:element>
```

另外，以下代码用于添加print_date属性：

```
<xsl:attribute name="print_date">1998-02-07</xsl:attribute>
```

它们将产生以下XML结果：

```
<title print_date="1998-02-07">Instant JavaScript</title>
```

E.1.2 XSL样式表运行时方法

通过xsl:eval元素，能够执行 IE5的XSL中内置的若干方法。IXTLRuntime对象提供了以下方法：

表 E-2

名 称	描 述
absoluteChildNumber (this_node)	返回指定节点在它的父节点的 childNodes列表中的索引 值从“1”开始
ancestorChildNumber(node_ name, this_node)	从指定节点的祖先节点中寻找具有指定名称的第一个祖先节点，并 返回该节点在它的父节点的 childNodes列表中的索引。值从“1”开 始。如果没有符合条件的祖先节点，则返回0
childNumber (this_node)	利用指定节点的父节点的 childNodes列表，构造与指定节点的名称 相同的子节点列表，并返回指定节点在该列表中的索引（即：返回 指定节点在同名的兄弟节点列表中的索引），如果找不到同名的兄弟 节点，则返回0。值从“1”开始
depth(start_node)	返回指定节点在文档树中所处的深度或层次。XMLDocument或根 节点位于第0层
elementIndexList (this_node, node_name)	返回指定节点以及直至文档根节点（包括根节点在内）的 所有祖先节点的索引数组，它用于表示每个节点在它的父节点的 childNodes列表中的位置。数组从文档根节点开始排列 如果不指定 node_name参数，本方法将返回表示索引的整数数组， 例如：指定节点相对于它的所有兄弟节点的索引，指定节点的父节 点相对于相应的兄弟节点的索引，依此类推，直至到达文档的根节 点 如果指定了 node_name参数，返回的数组中仅包含具有指定名称的 节点项，并且索引是相对于具有指定名称的兄弟节点计算的。零表 示在树的该层中没有具有指定名称的节点 虽然微软的文档中包含本方法，但是在编写本书时，IE5并不支持它
formatDate(date, format, locale)	使用指定的格式选项设定日期参数值的格式。它支持以下格式编码： m——月（1-12） mm——月（01-12） mmm——月（Jan-Dec） mmmm——月（January-December） d——日（1-31） dd——日（01-31） ddd——日（Sun-Sat） dddd——日（Sunday-Saturday） yy——年（00-99） yyyy——年（1900-1999） 地区用于确定日期值的正确顺序。如果省略该参数，则缺省的顺序 为月-日-年
formatIndex (number, format)	使用指定的数字系统设定整数的格式 1——标准计数系统 01——标准计数系统，有前导的零

(续)

名 称	描 述
	A——大写字母序列：“A”到“Z”，然后是“AA”到“ZZ” a——小写字母序列：“a”到“z”，然后是“aa”到“zz” I——大写罗马数字：“I”，“II”，“III”，“IV”，等等 i——小写罗马数字：“i”，“ii”，“iii”，“iv”，等等
formatNumber (number, format)	使用指定的格式设定值数字的格式。格式字符串中可以包含零或者多个下面列出的值： #（磅）——只显示有意义的数字，忽略没有意义的零 0（零）——在这些位置显示没有意义的零 ?（问号）——在小数点的任一端添加没有意义的零，使得小数点对齐。你还可以将该符号用于长度可变的小数部分 .（句点）——指示小数点的位置 ,（逗号）——显示千位分隔符，或者以千为单位分隔数字 %（百分号）——将数字显示为百分数 E或e——以科学计数法（指数）格式显示数字。如果格式中指数码右侧包含零或#，则以科学计数法显示数字，并插入“E”或“e”。其中0或#的字符数决定了指数的位数 E-或e-——在负指数的前面增加减号 E+或e+——在负指数的前面增加减号，在正指数的前面增加加号
formatTime(time, format, locale)	使用指定的格式选项设定时间参数值的格式。它支持以下格式编码： h——时（0-23） hh——时（00-23） m——分（0-59） mm——分（00-59） s——秒（0-59） ss——秒（00-59） AM/PM——增加“AM”或“PM”，并以12小时格式显示时间 am/pm——增加“am”或“pm”，并以12小时格式显示时间 A/P——增加“A”或“P”，并以12小时格式显示时间 a/p——增加“a”或“p”，并以12小时格式显示时间 [h]:mm——以小时为单位显示经过的时间，例如：“25.02” [mm]:ss——以分为单位显示经过的时间，例如：“63:46” [ss]——以秒为单位显示经过的时间 ss.00——显示秒的小数部分 地区用于确定正确的分隔字符
uniqueID(this_node)	返回指定节点的唯一标识符

下面的代码使用内置的formatIndex()方法将当前元素的内容（数字）转换为罗马数字：

```
<xsl:eval>  
  intNumber=parseInt(this.text);  
  formatIndex(intNumber, "i");  
</xsl:eval>
```

需要注意的是，元素的内容首先要转换为字符串格式（这是所有XML内容的缺省格式，除非我们在XML文档的模式中指定其它数据类型）。

E.2 IE5 XSL模式匹配语法

利用前面描述的元素，XSL能够创建包含一个或多个XSL template元素的样式表文档。这些模板将应用于源文档中的一个元素或一组元素，并创建输出文档的特定部分。为了定义将哪个模板应用于哪个源元素或节点，要使用模式（pattern）。模式可以采用以下两种通用形式之一，并通过以下方式定义与之匹配的一个或多个节点：

- 一个或多个节点在源文档中的位置和层次。
- 利用过滤器有选择性地寻找目标节点。

E.2.1 节点位置和层次

为了通过节点（即：元素）在源文档中的位置和层次选择或匹配节点，我们使用一连串的路径运算符（path operator）建立模式字符串。下表列出了IE5 XSL支持的路径运算符：

表 E-3

运算符	描 述
/	斜杠代表子路径运算符。它选择指定节点的直接子元素，其作用类似于在URL中指定路径。例如，我们使用book/category选择<book>元素的子元素中所有<category>元素。为了表示根节点，我们将该运算符放在模式串的开始处，例如： /booklist/book
//	双斜杠代表递归子孙路径运算符。它选择当前节点下任意深度级的所有匹配节点（所有匹配的子孙节点），例如：booklist//title用于选择<booklist>元素的任一级子孙节点中的所有<title>元素。当它出现在模式串的开始处时，表示从根节点开始递归，即：文档中的所有元素
.	句点代表当前上下文路径运算符。它用于表示当前节点或“上下文”，例如：./title用于选择位于当前节点之下任一级的所有<title>元素。运算符组合./表示当前上下文，它通常是多余的——例如：./book/category等同于book/category
@	“at”运算符代表属性路径运算符。它表示模式串的这部分是指当前元素的属性。例如：book/@print_date用于选择所有<book>元素的print_date属性
*	星号代表通配路径运算符，它用于选择所有元素或属性（不考虑这些元素或属性的名称），例如：book/*用于选择所有<book>元素的所有子元素，book/@*用于选择所有<book>元素的所有属性

节点索引位置

路径运算符将返回与模式匹配的所有元素或节点。节点索引能够在匹配的节点集合中指定特定的节点，特殊的XSL end()函数用于指定最后一个节点：

```
/booklist/book[0]      根元素<booklist>的第一个<book>子元素
/booklist/book[2]      根元素<booklist>的第三个<book>子元素
/booklist/book[end()]  根元素<booklist>的最后一个<book>子元素
```

需要注意的是，以下三个例子从同一文档中分别选取三个不同的节点：

```
book/category[2]      所有<book>元素的第二个<category>子元素
book[2]/category[2]    第二个<book>元素的第二个<category>子元素
(book/category)[2]     所有<book>元素的<category>子元素集合中的第二个
```

'<category>元素

在最后一个例子中，首先根据圆括号中的模式串创建所有 <book>元素的所有<category>子元素集合，然后根据索引运算符从中选择第二个元素。

E.2.2 XML过滤器和过滤模式

XML过滤器的通用格式为 [operator pattern]，其中operator是可选的过滤运算符，它定义了如何应用模式，pattern是必需的XML过滤模式，它用于根据特定的标准范围选择一个或多个元素。过滤运算符和过滤模式之间以一个或多个空格分隔。如果需要的话，可选的 operator部分能够包含多个过滤运算符表达式。如果省略过滤运算符，与过滤模式中的标准匹配的所有节点都将被选中。

1. 过滤模式

XSL过滤模式的功能非常强大，它几乎能够提供无限的模式组合。下面我们将分几类介绍模式的应用：

- 根据子节点名称选择。
- 根据节点值选择。
- 根据属性存在性选择。
- 根据属性值选择。
- 根据以上方法的组合选择。

1) 根据子节点名称选择

我们前面介绍的位置和层次语法是根据元素的名称及它们在文档中的位置选择匹配的元素。例如：book/category用于选择<book>元素的所有<category>子元素。它等价于以下过滤器：

```
book[category]/category
```

过滤器book/category实际上是一种速记方式，它说明我们要选择所有含 <category>元素的<book>元素（等价于book[category]），然后选择<category>元素。当你希望返回另一个子元素时，会发现这种较长的写法比较有效。例如：

```
book[title]/category
```

以上模式意味着对于含<title>子元素的<book>元素，返回它的<category>子元素。如果要查找所有同时包含<category>和<title>子元素的<book>元素，可以使用双过滤器：

```
book[title][category]
```

2) 根据节点值选择

除了根据名称选择节点之外，我们还可以扩展过滤模式，使之根据值选择节点：

```
book[category = 'Scripting']
```

以上模式将选择包含<category>子元素且该子元素的值为‘Scripting’的所有<book>元素。如果我们希望得到这类书的书名，可以使用：

```
book[category = 'Scripting']/title
```

为了指定当前元素的值，可以使用句点路径运算符。例如：

```
book/title[. = 'Instant JavaScript']
```

以上模式将选择图书 ‘ Instant JavaScript ’ 的书名。

3) 根据属性存在性选择

过滤模式可以使用 ‘ @ ’ 属性运算符指定元素必须有匹配的属性：

```
book[@print_date]
```

以上模式仅选择有 print_date 属性的 book 元素。

4) 根据属性值选择

我们还可以在模式中指定属性的值：

```
book[@print_date = '1998-05-02']
```

5) 根据以上方法的组合选择

当然，我们还可以将以上方法组合在一起，对元素或节点进行更精确的选择。例如：

```
book[@print_date = '1998-05-02']/title[. = 'Instant JavaScript']
```

以上模式用于查找书名为 ‘ Instant JavaScript ’ 且在 1998 年 5 月 2 日出版的书。

```
/booklist//cover_design[issue = "final"]/*[@url = 'images']
```

以上模式用于选择符合下列条件的所有元素

- 它的名称随意，但是有名为 url 的属性，且属性值为 ‘ images ’（根据 *[@url = 'images']）
- 它是 cover_design 的子元素，且 cover_design 本身有值为 ‘ final ’ 的子元素 issue（根据 cover_design[issue = "final"]）
- 它是根元素 booklist 的子孙元素（根据 /booklist//）

需要注意的是，元素和属性的值可以包含在单引号或双引号中。

2. 比较运算符

以上例子都使用等于运算符 ‘ = ’ 判断两个值是否相等。等号可以用于数字和字符串。缺省情况下，所有 XML 值都是字符串，但是在进行比较时，IE5 会尽可能将它们转化为适当的数据类型。数据类型是根据节点值字符串的内容或者用于指定数据类型的模式（ schema ）选择的（如果存在模式的话）。这意味着比较运算 [price = 29.95]（数字值不带引号）是有效的。

提示 如果存在模式，且节点的内容不能转化为模式中指定的类型，例如：它所包含的字符对于该数据类型来说是不合法的，比如数字值中的字母，它将不作为匹配的节点。

除了等于运算符，IE5 XSL 还提供其它比较运算符：

表 E-4

简单表示法	运算符	描 述
=	\$eq\$	区分大小写的等于，例如： [price = 29.95]
!=	\$ne\$	区分大小写的不等于，例如： [category != 'Script']
<*	\$lt\$	区分大小写的小于，例如： [radius \$lt\$ 14.73]

(续)

简单表示法	运算符	描 述
<=*	\$le\$	区分大小写的小于等于, 例如: [age \$le\$ 18]
>	\$gt\$	区分大小写的大于, 例如: [name > 'H']
>=	\$ge\$	区分大小写的大于等于, 例如: [speed >= 55]
	\$ieq\$	不区分大小写的等于
	\$ine\$	不区分大小写的不等于
	\$ilt\$	不区分大小写的小于
	\$ile\$	不区分大小写的小于等于
	\$igt\$	不区分大小写的大于
	\$ige\$	不区分大小写的大于等于

注: XSL属性中不能包含 '`<`' 和 '`<=`' 运算符, 因为属性必须符合 XML 标准的格式正规约束。因此, 最好使用等价的 `lt` 和 `le`。另外, 所有过滤运算符名称 (例如: `eq`) 都区分大小写, 因此它们必须是小写的。

运算符的简单表示法和较长的表示法使用方式相同, 因此以下两个模式串是等价的:

```
[category = 'Scripting']
[category $eq$ 'Scripting']
```

同样, 以下两个模式串也是等价的:

```
[category != 'Scripting']
[category $ne$ 'Scripting']
```

不区分大小写的运算符没有对应的简单表示法。然而, 对于不需要考虑大小写的匹配, 它非常有用。XSL中没有UCase或LCase函数 (除非你自己编写相应的脚本函数), 因此它能够避免多次匹配, 例如:

```
[category = 'html' $or$ category = 'HTML']
```

可以直接改为:

```
[category $ieq$ 'html']
```

3. 逻辑过滤运算符

除了比较运算, 我们还可以在模式中增加逻辑运算符, 产生更加复杂的模式串 (如同前一节的最后一个例子)。下表列出了所有逻辑运算符:

表 E-5

简单表示法	运 算 符	描 述
&&	\$and\$	逻辑与 (AND)
	\$or\$	逻辑或 (OR)
	\$not\$	逻辑非 (NOT)

因此, 利用以上运算符, 我们可以选择 `<category>` 元素等于 'Scripting' 或 'HTML' 的书:

```
book/[category = 'Scripting' $or$ category = 'HTML']
```

我们还可以选择书名为 ‘ Instant JavaScript ’ (不区分大小写的匹配), 但不属于 ‘ Scripting ’ 类的图书 :

```
book/[category $ne$ 'Scripting' $and$ title $ieq$ 'Instant JavaScript']
```

\$not\$ 仅仅改变匹配的 “ 事实 ”, 因此以下两个模式串是等价的, 它们都匹配有值为 ‘ Scripting ’ 的子元素 <category>, 且没有值为 ‘ HTML ’ 的子元素 <category> 的 <book> 元素 (因此它不包括同时有这两个值的 <category> 子元素的 <book> 元素) :

```
book/[category = 'Scripting' $and$ category $ne$ 'HTML']
```

```
book/[category = 'Scripting' $and$ $not$ category = 'HTML']
```

4. 过滤集合运算符

在以上所有使用比较运算符的过滤模式例子中, 都是以缺省的过滤操作为基础的, 即: 如果过滤器中没有指定运算符, 则返回匹配模式的任何或所有节点。然而, 我们能够更加精确地指定所需的匹配元素, 它类似于使用索引指定第一个元素。为此, 我们将使用集合运算符 \$any\$ 和 \$all\$:

表 E-6

运 算 符	描 述
\$all\$	仅当指定的模式匹配集合中的所有项时返回 True。
\$any\$	如果指定的模式匹配集合中的任何一项, 返回 True。

为了理解这两者之间的差别, 最简单的方法是考虑元素是如何被挑选的。对于名为 <book> 的元素, 我们可以使用以下模式指定当它有为 ‘ HTML ’ 的子元素 <category> 时它被包含:

```
book[category = 'HTML']
```

然而, 它仅仅匹配第一个 <category> 子元素的值为 ‘ HTML ’ 的 <book> 元素。如果第一个 <category> 的值不是 ‘ HTML ’, 即使其它 (后续的) 子元素满足这个条件, <book> 元素也不会被选中。然而, 如果我们使用模式:

```
book[$any$ category = 'HTML']
```

我们将得到所需的 <book> 元素, 因为我们指定希望匹配任何有为 ‘ HTML ’ 的 <category> 子元素的 <book> 元素。如果使用集合运算符 \$all\$, 我们将指定选出所有 <category> 子元素的值都为 ‘ HTML ’ 的 <book> 元素, 而不仅仅是第一个或任何一个子元素符合条件的 <book> 元素。由此得到的结果中的 <category> 子元素的值都将是 ‘ HTML ’:

```
book[$all$ category = 'HTML']
```

当然, 如果某个 <book> 元素只有一个 <category> 子元素, 且值为 ‘ HTML ’, 则这三个过滤器都将返回这个元素。仅当模式指定的资源有多个匹配的子元素 (或其他元素) 时, 才会体现出这三者的差别。

E.2.3 XSL 内置的方法

当我们在前面介绍如何根据索引选择元素时, 曾经看到一个内置的 XSL 方法。end() 方法将

返回匹配的节点集合中的最后一个节点：

```
booklist/category[end()]
```

1. 信息方法

信息方法有助于从集合中分离出特定的节点。

表 E-7

名 称	描 述
end()	选择并返回集合中的最后一个节点
index()	选择并返回当前节点在集合中的索引（号）
nodeName()	选择并返回当前节点的标记名称，其中包含任何命名空间前缀
nodeType()	选择并返回代表节点类型的数字（类似于 DOM 中相应的方法）
date()	返回日期格式的值
text()	选择并返回当前节点的文本内容
value()	返回当前节点的值，并转化为适当的类型

value()方法是缺省的，因此以下两个模式串是等价的：

```
book[category!value() = "Scripting"]
```

```
book[category = "Scripting"]
```

提示 叹号运算符（有时称为‘bang’运算符）表示value()是<category>元素的方法。常见的句点用法是不合法的。它有可能与当前路径运算符混淆。

当我们需要获得特定的元素时，index()方法也是可选的：

```
book[index() = 5]
```

```
book[5]
```

然而，index()方法可以用于选择几个元素，例如：仅选择第四个和第五个 <book>元素：

```
book[index() > 3 $and$ index() < 6]
```

2. 集合方法

另外，IE5支持的XSL的集合方法也可以用于选择元素或其他节点：

表 E-8

名 称	描 述
ancestor()	选择与模式匹配的，且与当前节点距离最近的祖先节点，它从父节点开始，沿着文档层次结构向上搜索。它返回一个元素，如果不存在匹配的元素，则返回 null
attribute()	选择当前节点的所有属性节点，并作为一个集合返回。可以通过参数指定要匹配的属性名称
comment()	选择所有子注释节点，并作为一个集合返回
element()	选择当前节点的所有子元素节点，并作为一个集合返回。可以通过参数指定要匹配的元素名称
node()	选择所有非属性子节点，并作为一个集合返回
pi()	选择所有子处理指令节点，并作为一个集合返回
textnode()	选择所有子文本节点，并作为一个集合返回

我们将使用下面的模式串选择 <book>元素的所有注释元素：

```
book/comment()
```

attribute()和element()方法可以包含用于限制匹配节点的文本参数：

```
book/attribute('print_date')
```

当然，它等价于我们在前面介绍的 ‘@’ 运算符，因此以下两个模式串的结果是相同的：

```
book/attribute('print_date')
```

```
book/@print_date
```

element()方法也有等价的表示法，以下两个模式串的结果是相同的：

```
book/element('category')
```

```
book/category
```

ancestor()方法的文本参数中可以包含要匹配的模式。例如：

```
ancestor(book/category)
```

该方法将匹配最近的<category>祖先节点，且该节点是<book>元素的子节点。需要注意的是，该方法不能出现在模式串中 ‘/’ 或 ‘//’ 的右侧，而且与 attribute()和element()方法不同的是，要匹配的节点名称不能包含在引号中。

E.2.4 重要说明

值得注意的是，在所有与 XML相关的技术中，XSL可能是目前最不稳定的，它的语言和语法都可能发生变化。W3C的草案与IE5的XSL实现之间存在着微妙的差别。目前，你的开发工作应该定位在试验阶段，直至将来出现更加可靠的标准。