XSLT 元素的使用说明

原始作者	中文整理人	整理时间	版本	说明
W3C school	skater	2005-8-17	0.1	XSLT 初稿
说明 W3C school 代码 互联网 MSDN			0.2	增加函数说明
互联网		2005-8-21	0.21	增加运算符说明

前言

这段时间在看有关 XML 方面的东西，苦于网上找不到比较全面的中文教程，于是去 W3C 找到相关英文资料读，想到以后的朋友可能也会碰到这样的问题，于是把学到的东西整理与此。本文档系翻译 W3C School 上的相关文章，翻译中很多部分不是很准确，希望大家能帮我指出来。另外本文档会一直更新，我会陆续把其他的 XML 使用说明放上来。本文档中所有代码，本人都调试过，都可以正常使用。

本文档版权归原作者所有。
在免费、且无任何附加条件的前提下，可在网络媒体中自由传播。
如需部分或者全文引用，请事先征求作者意见。

如果本文对您有些许帮助，表达谢意的最好方式，是将您发现的问题和文档改进意见及时反馈给作者。当然，倘若有时间和能力，能为技术群体无偿贡献自己的所学为最好的回馈。（呵呵，这段是拷贝夏昕的话，希望夏昕别在意）。

我的邮箱是xqflying@163.com

skater

2005-8-21

目录

XSLT 元素

<xsl:apply-imports> 元素

定义与用法
<xsl:apply-imports>运用了一个从外部导入的 xsl 作为 Template.
导入的 Template 的样式表比宿主 xsl 的样式表优先级要低.

语法
<xsl:apply-imports/>

Attributes
None

例子
假设我们有一个样式表叫"standard.xsl"，它包含一个为 message elements 定义的 template

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="message">
  <h2>
<xsl:apply-templates/>
</h2>
</xsl:template>
</xsl:stylesheet>
```

宿主样式表要能导入"standard.xsl"，并且修改 message elements，像如下这样：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:import

href="standard.xsl"/><xsl:template match="message">
   <div style="border:solid blue">
   <xsl:apply-imports/>
   </div>
</xsl:template></xsl:stylesheet>
```

结果会把 message elements 转变成如下这样：

```
<div style="border:solid blue"><h2>...</h2></div>
```

用这种方式，可以实现 xsl 的重用。

<xsl:apply-templates> 元素

定义与用法

<xsl:apply-templates>元素将模版运用到当前元素或者是当前元素的子节点。

如果我们在<xsl:apply-templates>元素中增加一个 select 属性，它将只对与属性匹配的子元素有效。我们可以用 select 属性来指定要处理的子节点。

语法

```
<xsl:apply-templates select="expression" mode="name">
<!-- Content:(xsl:sort|xsl:with-param)* -->
</xsl:apply-templates>
```

属性

属性	值	说明
select	expression	select 表达式是可选的用于指定要处理的节点。一个星号*选择了全部的节点集。如果属性省略了，那么所有的子节点都将被选择。
mode	name	mode 也是可选的，如果对一个相同的元素有很多定义，那么用 mode 可以区分他们。

例子 1

对于文档中的每一个 tiltle 元素用 h1 元素包装。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="title">
  <h1><xsl:apply-templates/></h1>
</xsl:template>
</xsl:stylesheet>
```

例子 2
对于文档中的所有 message 元素的子元素 tiltle 用 h1 元素包装。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="message">
  <h1><xsl:apply-templates select="title"/></h1>
</xsl:template>
</xsl:stylesheet>
```

例子 3
对于文档中的所有 message 元素的所有子元素用 h1 元素包装，mode 属性被置为”big”。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="message">
  <h1><xsl:apply-templates select="*" mode="big"/></h1>
</xsl:template>
</xsl:stylesheet>
```

<xsl:attribute>元素
定义与用法
<xsl:attribute>元素被用来向元素添加属性

说明：当有相同的名称时<xsl:attribute>元素会替代当前的属性。

语法

```
<xsl:attribute name="attributename" namespace="uri">
<!-- Content:template -->
</xsl:attribute>
```

属性

属性	值	说明
name	attributename	必需的。指定属性的名称
namespace	URI	可选。为属性定义命名空间。

例子 1
在 picture 元素里添加一个 source 属性：

```
<picture>
   <xsl:attribute name="source"/>
</picture>
```

例子 2
在 picture 元素里添加一个 source 属性，并用"images/name"付值：

```
<picture>
   <xsl:attribute name="source">
      <xsl:value-of select="images/name" />
   </xsl:attribute>
</picture>
```

<xsl:attribute-set> 元素

定义与用法

<xsl:attribute-set>元素创建一个属性集。

<xsl:attribute-set>
说明：必须是<xsl:stylesheet> 或者 <xsl:transform>的子元素。

语法

```
<xsl:attribute-set name="name" use-attribute-sets="name-list">
   <!-- Content:xsl:attribute* -->
</xsl:attribute-set>
```

属性

属性	值	说明
name	name	必需。用来指定 attribute-set 的名称。

use-attribute-sets	name-list	可选。

例子 1
建立一个可以应用于任何输出元素的属性集：

```
<xsl:attribute-set name="font">
   <xsl:attribute name="fname">Arial</xsl:attribute>
   <xsl:attribute name="size">14px</xsl:attribute>
   <xsl:attribute name="color">red</xsl:attribute>
</xsl:attribute-set>
```

<xsl:call-template>元素
定义与用法
<xsl:call-template>元素调用一个命名的 template。

语法

```
<xsl:call-template name="templatename">
<!-- Content:xsl:with-param* -->
</xsl:call-template>
```

属性

属性	值	说明
name	templatename	必须。 指定被调用的 template 的名字

例子 1
当发现 car 元素时，调用名为"description"的 template

```
<xsl:template match="car">
   <xsl:call-template name="description"/>
</xsl:template>
```

<xsl:choose>元素

定义与用法
与<xsl:when> 和<xsl:otherwise>结合使用，表示多条件选择。

当<xsl:when>不为真时，<xsl:otherwise>的内容就被执行。

当<xsl:when>和<xsl:otherwise>都不为真时，什么都不执行。

说明：对于简单的条件选择，用<xsl:if>即可。

语法
<xsl:choose><!-- Content:(xsl:when+,xsl:otherwise?) --></xsl:choose>

属性
无

例子 1
下面的代码展示了如何将价格高于 10 的 CD 的 artist 列背景颜色设为粉红色

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <xsl:choose>
          <xsl:when test="price &gt; 10">
            <td bgcolor="#ff00ff">
            <xsl:value-of select="artist"/>
            </td>
          </xsl:when>
          <xsl:otherwise>
            <td><xsl:value-of select="artist"/></td>
          </xsl:otherwise>
        </xsl:choose>
      </tr>
      </xsl:for-each>
    </table>
```

```
  </body>
  </html>
</xsl:template></xsl:stylesheet>
```

例子 2
声明一个名为"color"的变量。把当前元素 color 属性的值赋给"color"变量。如果当前元素没有 color 属性，则"color"变量的值将为"green"：

```
<xsl:variable name="color">
  <xsl:choose>
    <xsl:when test="@color">
      <xsl:value-of select="@color"/>
    </xsl:when>
    <xsl:otherwise>green</xsl:otherwise>
  </xsl:choose>
</xsl:variable>
```

<xsl:comment> 元素

<xsl:comment>元素
定义和用法
<xsl:comment>元素被用来在结果树中创建一个注释节点。

语法

```
<xsl:comment><!-- Content:template --></xsl:comment>
```

属性
无

例子 1

```
<xsl:comment>This is a comment!</xsl:comment>
```

<xsl:copy> 元素

定义与用法
<xsl:copy>创建当前节点的一份拷贝

说明：当前节点的命称空间被自动拷贝，但是子节点和当前节点的属性不会被自动拷贝！

语法
<xsl:copy use-attribute-sets="name-list"> <!-- Content:template --></xsl:copy>

属性

属性	值	说明
use-attribute-sets	name-list	可选。

例子 1
把 message 节点拷贝到输出文档中：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="message">
  <xsl:copy>
    <xsl:apply-templates/>
  </xsl:copy>
</xsl:template>

</xsl:stylesheet>
```

<xsl:copy-of> 元素
定义与用法
<xsl:copy-of>元素创建当前节点的一份拷贝。

注意：命名空间节点，自节点和当前节点的属性都会被自动拷贝。

提示：这个元素能够向输出的不同位置插入多个相同节点的拷贝。

语法
<xsl:copy-of select="expression"/>

属性

属性	值	说明
select	expression	需要，用来指定要拷贝的部分

例子 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:variable name="header">
  <tr>
  <th>Element</th>
  <th>Description</th>
  </tr>
</xsl:variable>
<xsl:template match="/">
  <html>
  <body>
  <table>
    <xsl:copy-of select="$header" />
    <xsl:for-each select="reference/record">
    <tr>
    <xsl:if test="category='XML'">
      <td><xsl:value-of select="element"/></td>
      <td><xsl:value-of select="description"/></td>
    </xsl:if>
    </tr>
    </xsl:for-each>
  </table>
  <br />
  <table>
    <xsl:copy-of select="$header" />
    <xsl:for-each select="table/record">
    <tr>
    <xsl:if test="category='XSL'">
      <td><xsl:value-of select="element"/></td>
      <td><xsl:value-of select="description"/></td>
    </xsl:if>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

<xsl:decimal-format>元素

定义与用法

<xsl:decimal-format> 元素定义字符和符号使用 format-number()函数来把数字转换为字符串。

每个国家使用不同的字符分割整数部分，以及整数分组符号。<xsl:decimal-format>元素让您可以自己定义它们。
用<xsl:decimal-format>元素您可以把特殊字符转为其他的符号。

这个元素是个顶级元素。

format-number()函数可以用 name 指定<xsl:decimal-format>元素。

语法
<xsl:decimal-format
name="name"
decimal-separator="char"
grouping-separator="char"
infinity="string"
minus-sign="char"
NaN="string"
percent="char"
per-mille="char"
zero-digit="char"
digit="char"
pattern-separator="char"/>

属性

属性	值	说明
name	name	可选。指定这个 format 的 name
decimal-separator	char	可选。指定小数点，默认是“.”
grouping-separator	char	可选。指定千分割符，默认是“,”(比如 2,000)
infinity	string	可选。指定 “无穷大”字符串，默认是“Infinity”
minus-sign	char	可选。指定负数符号。默认是“-”
NaN	string	可选。当值不是数字时，指定使用的字符串。默认是“NaN”
percent	char	可选。指定百分号，默认是“%”

per-mille	char	可选。指定千分号，默认是“ ‰ ”
zero-digit	char	可选。指定零。默认是“0”
digit	char	可选。指定字符用来指出需要使用数字的地方。默认是#(看下面的例子您会明白)
pattern-separator	char	可选。指定字符用来在一个样式中分割正数和负数，默认是“;”

例子 1
下面的这个例子展示了如何格式化为欧洲流通样式。（注意在 format-number()中的第三个变量运用了定义的<xsl:decimal-format>的名称）:
（PS：欧洲使用这种标准？我也是才知道）
下面这个例子用“.”作为千分割符，用“;”作为小数点。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:decimal-format name="euro" decimal-separator="," grouping-separator="."/>
<xsl:template match="/">

<xsl:value-of select="format-number(26825.8, '#.###,00', 'euro')"/>

</xsl:template>
</xsl:stylesheet>
```

输出: 26.825,80

<xsl:element>元素

定义和用法
<xsl:element>元素用来创建一个元素到输出文档。

语法

```
<xsl:element
name="name"
namespace="URI"
use-attribute-sets="namelist">  <!-- Content:template --></xsl:element>
```

属性

属性	值	说明
name	name	必须。指定创建的元素的名称（元素名称可以动态赋予，比如 <xsl:element name="{$country}" />）
namespace	URI	可选。指定元素的名称空间（可以动态赋予，比如：<xsl:element name="{$country}" namespace = "{$someuri}"/>）
use-attribute-sets	namelist	可选。指定添加的属性列表。

例子 1

创建一个名为“singer”的元素，此元素包含每一个 artist 元素的值。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <xsl:for-each select="catalog/cd">
    <xsl:element name="singer">
      <xsl:value-of select="artist" />
    </xsl:element>
    <br />
  </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

<xsl:fallback>元素

定义和用法
The <xsl:fallback> 元素，当 XSL 处理器不能处理某个 XSL 元素时，<xsl:fallback>元素指定了一个替换代码去执行。

语法
<xsl:fallback><!-- Content: template --></xsl:fallback>

属性
无

例子 1

这个例子假设处理器对<xsl:loop>元素不支持，它将会用<:xsl:for-each>元素替代。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="catalog/cd">
  <xsl:loop select="title">
    <xsl:fallback>
      <xsl:for-each select="title">
        <xsl:value-of select="."/>
      </xsl:for-each>
    </xsl:fallback>
  </xsl:loop>
</xsl:template>
</xsl:stylesheet>
```

<xsl:for-each> 元素

定义与用法
<xsl:for-each>元素在每一个指定的节点集中循环。

语法

```
<xsl:for-each select="expression">
<!-- Content:(xsl:sort*,template) -->
</xsl:for-each>
```

属性

属性	值	说明
select	expression	必须。指定要处理的节点。

例子 1
在每个 cd 节点中循环，使用<xsl:value-of>把每一个 cd 节点中的 title 和 artist 的节点值写到输出中：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

例子 2

在每个 cd 节点中循环，使用<xsl:value-of>把每一个 cd 节点中的 title 和 artist 的节点值写到输出中（用 artist 作为排序条件）：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <xsl:sort select="artist"/>
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
```

```
        </tr>
        </xsl:for-each>
      </table>
    </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

<xsl:if>元素

定义和用法
<xsl:if>元素包含一个样规，该样规当指定的条件为真时应用。

提示：可以使用<xsl:choose>与<xsl:when> ， <xsl:otherwise>结合，表达多条件控制！

语法

```
<xsl:if test="expression">
<!-- Content: template -->
</xsl:if>
```

属性

属性	值	说明
test	expression	必须。指定要被检查的条件。

例子 1
选择 CD 中 price 的值大于 10 的。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
    <html>
    <body>
      <h2>My CD Collection</h2>
      <table border="1">
        <tr bgcolor="#9acd32">
          <th>Title</th>
          <th>Artist</th>
```

```
        </tr>
        <xsl:for-each select="catalog/cd">
        <xsl:if test="price &gt; 10">            <!—注意  &gt;   代表大于  -->
          <tr>
            <td><xsl:value-of select="title"/></td>
            <td><xsl:value-of select="artist"/></td>
          </tr>
        </xsl:if>
        </xsl:for-each>
      </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

例子 2

展示每一个 CD 的 title。

1、 在每一个 CD-title 后插入“,”（条件：不是最后一个）

2、 如果是倒数第二个插入“, and”

3、 如果是最后一个插入“!”

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <p>Titles:
    <xsl:for-each select="catalog/cd">
      <xsl:value-of select="title"/>
      <xsl:if test="position()!=last()">
        <xsl:text>, </xsl:text>
      </xsl:if>
      <xsl:if test="position()=last()-1">
        <xsl:text> and </xsl:text>
      </xsl:if>
      <xsl:if test="position()=last()">
        <xsl:text>!</xsl:text>
      </xsl:if>
    </xsl:for-each>
    </p>
  </body>
  </html>
</xsl:template>
```

</xsl:stylesheet>

<xsl:import> 元素

定义和用法
<xsl:import>元素是一个高级（原文是 top-level）的元素用来把一个样式表的内容导入到另外一个。

说明：一个被导入的样式表比导入它的样式表的优先级要低。

说明：这个元素必须在<xsl:stylesheet> 或<xsl:transform>元素中作为第一个子节点出现。

说明：在 Netscape 6 中不支持 import 优先级规则，所以这个元素在 Netscape 6 中与<xsl:include>是一样的。

语法
<xsl:import href="URI"/>

属性

属性	值	说明
href	URI	必须。指定要被导入的样式表的 URI

例子 1
假设你有一个称为"cdcatalog_ex3.xsl"的样式表。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <tr>
        <td><xsl:value-of select="catalog/cd/title"/></td>
```

```
        <td><xsl:value-of select="catalog/cd/artist"/></td>
       </tr>
    </table>
 </body>
 </html>
</xsl:template></xsl:stylesheet>
```

第二个称为"cdcatalog_import.xsl"的样式表导入"cdcatalog_ex3.xsl"：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:import href="cdcatalog_ex3.xsl"/>

<xsl:template match="/">

 <xsl:apply-imports/>

</xsl:template>
</xsl:stylesheet>
```

说明：这个例子不能运行在 Netscape 6 中，因为它不支持<xsl:apply-imports> 元素。

<xsl:include> 元素

定义和用法

<xsl:include>元素是一个高级（原文是 top-level）的元素用来把一个样式表的内容导入到另外一个。

说明：Included 的样式表与 including 的样式表有相同的优先级。

说明：这个元素必须作为<xsl:stylesheet> 或 <xsl:transform>的一个子节点出现。

语法

```
<xsl:include href="URI"/>
```

属性

属性	值	说明
href	URI	必须。指定 include 的样式表的 URI。

<xsl:key> 元素

定义与用法

<xsl:key>元素是一个高级（原文是 top-level）的元素，用来声明一个被 key()函数使用的 key

说明：一个 key 不一定是唯一的！

语法
<xsl:key name="name" match="pattern" use="expression"/>

属性

属性	值	说明
name	name	必须。指定 key 的名称。
match	pattern	必须。定义 key 将应用的节点。
use	expression	必须。Key 对于每一个节点的值。

例子 1
假设你有一个称为"persons.xml"的 XML 文件：

```
<persons>
   <person name="Tarzan" id="050676"/>
   <person name="Donald" id="070754"/>
   <person name="Dolly" id="231256"/>
</persons>
```

你可以在一个 XSL 文件中这样定义一个 key:

```
<xsl:key name="preg" match="person" use="@id"/>
```

查找 id="050676"的 person。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
<xsl:key name="preg" match="person" use="@id"/>

<xsl:template match="/">
  <html>
  <body>
  <xsl:for-each select="key('preg','050676')">
    <p>
    Id: <xsl:value-of select="@id"/><br />
    Name: <xsl:value-of select="@name"/>
    </p>
  </xsl:for-each>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

<xsl:message> 元素

定义与用法
<xsl:message>元素向输出发送一条消息。这个元素首先被用来报告错误。

这个元素能够包含几乎所有其它的 XSL 元素（比如：<xsl:text>， <xsl:value-of>等）。

当一个错误发生时，Terminate 属性为您提供选择是继续执行还是退出。

语法

```
<xsl:message terminate="yes|no">
  <!-- Content:template -->
</xsl:message>
```

属性

属性	值	说明
terminate	yes no	可选。当消息写到输出后“yes”将终止执行，而“no”继续执行。默认是“no”。

例子 1
检查 artist 是否是一个空串。如果是，我们结束 XSL 处理并显示一条消息：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <xsl:for-each select="catalog/cd">
    <p>Title: <xsl:value-of select="title"/><br />
    Artist:
    <xsl:if test="artist="">
      <xsl:message terminate="yes">
        Error: Artist is an empty string!
      </xsl:message>
    </xsl:if>
    <xsl:value-of select="artist"/>
    </p>
  </xsl:for-each>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

<xsl:namespace-alias> 元素

定义与用法

<xsl:namespace-alias>元素用来在输出中用不同的名称空间替代样式表中原来的名称空间。

说明：<xsl:namespace-alias>是一个高级的元素，必须是<xsl:stylesheet>或者<xsl:transform>的子节点。

语法

```
<xsl:namespace-alias
stylesheet-prefix="prefix|#default"
result-prefix="prefix|"#default"/>
```

属性

属性	值	说明
stylesheet-prefix	prefix #default	必须。指定您想要改变的名称空间。
result-prefix	prefix #default	必须．指定输出想要的名称空间。

例子 1

wxsl 前缀在输出中被转换为 xsl 前缀。

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:wxsl="http://www.w3schools.com/w3style.xsl">

<xsl:namespace-alias stylesheet-prefix="wxsl" result-prefix="xsl"/>

<xsl:template match="/">
  <wxsl:stylesheet>
    <xsl:apply-templates/>
  </wxsl:stylesheet>
</xsl:template>

</xsl:stylesheet>
```

<xsl:number> 元素

定义与用法
<xsl:number>元素被用来确定当前节点中的整数位置。也被用来格式化一个数字。

语法

```
<xsl:number
count="expression"
level="single|multiple|any"
from="expression"
value="expression"
format="formatstring"
lang="languagecode"
letter-value="alphabetic|traditional"
grouping-separator="character"
```

grouping-size="number"/>

属性

Attribute	Value	Description
count	expression	可选. An XPath expression that specifies what nodes are to be counted 一个 XPath 表达式用来指定什么样的节点被计数。
level	single multiple any	可选。对数字串产生像目录树一样的分级。 • single （默认） • multiple （多级） • any (Netscape 6 不支持)
from	expression	可选。用一个 XPath 表达式指定从哪里开始计数。
value	expression	可选。由用户指定一个数字来产生数字序列。
format	formatstring	可选。 定义了输出的数字的样式。可以是下述之一： • format="1" results in 1 2 3 . . • format="01" results in 01 02 03 (not supported by Netscape 6) • format="a" results in a b c . . (not supported by Netscape 6) • format="A" results in A B C. . (not supported by Netscape 6) • format="i" results in i ii iii iv . . (not supported by Netscape 6) • format="I" results in I II III IV . . (not supported by Netscape 6)
lang	languagecode	可选。指定语言的字母表。（ps:有疑问） Netscape 6 不支持。
letter-value	alphabetic traditional	可选。指定数字化时是用字母表还是传统。默认是字母表。（ps:有疑问）
grouping-separator	character	可选，指定数字组的分割符，默认是","
grouping-size	number	可选。指定多少个数字为一组，使用一个分割符，默认是 3

例子 1
<xsl:number value="250000" grouping-separator="."/>

输出：
250.000

例子 2

<xsl:number value="250000" grouping-size="2"/>

输出：
25,00,00

例子 3
<xsl:number value="12" grouping-size="1"
grouping-separator="#" format="I"/>

输出：
X#I#I

例子 4

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
  <html>
  <body>
  <p>
  <xsl:for-each select="catalog/cd">
    <xsl:number value="position()" format="1" />
    <xsl:value-of select="title" /><br />
  </xsl:for-each>
  </p>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

<xsl:otherwise> 元素

定义与用法

<xsl:otherwise>元素为<xsl:choose>元素指定一个默认的动作。这个动作当<xsl:when>的条件不满足的时候发生。

语法
<xsl:otherwise>

```
<!-- Content:template -->
</xsl:otherwise>
```

属性
无

例子 1
下面这段代码将把 cd 的 price 高于 10 的 artist 列的背景色设为粉红色，OTHERWISE 只是打印 artist 的 name：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <xsl:choose>
          <xsl:when test="price&gt;'10'">
            <td bgcolor="#ff00ff">
            <xsl:value-of select="artist"/></td>
          </xsl:when>
          <xsl:otherwise>
            <td><xsl:value-of select="artist"/></td>
          </xsl:otherwise>
        </xsl:choose>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template></xsl:stylesheet>
```

例子 2
声明一个名为"color"的变量。设置变量的值为当前元素的 color 属性的值。如果当前元素没有 color 属性，"color"的值将为"green"：

```
<xsl:variable name="color">
```

```
  <xsl:choose>
    <xsl:when test="@color">
      <xsl:value-of select="@color"/>
    </xsl:when>
    <xsl:otherwise>green</xsl:otherwise>
  </xsl:choose>
</xsl:variable>
```

<xsl:output> 元素

定义与用法

<xsl:output>元素定义了输出文档的格式：

说明：<xsl:output>是一个高级的（top level）元素，必须作为<xsl:stylesheet>或<xsl:transform>的子节点出现：

语法

```
<xsl:output
method="xml|html|text|name"
version="string"
encoding="string"
omit-xml-declaration="yes|no"
standalone="yes|no"
doctype-public="string"
doctype-system="string"
cdata-section-elements="namelist"
indent="yes|no"
media-type="string"/>
```

属性

属性	值	说明
method	xml html text name	可选。定义了输出的格式。默认是 XML（如果根节点的第一个子节点是<html>并且没有其他高优先级的文本节点，那么默认是 HTML） Netscape 6 只支持"html" 和 "xml"
version	string	可选。 设置版本。当用 method="html" 或者

		method="xml"时才用到。
encoding	string	可选。设置输出的编码。
omit-xml-declaration	yes no	可选。如果为"yes"，则 XML 声明（<?xml...?>）在输出中会被省略，否则就会加入。默认是"no"
standalone	yes no	可选。"yes"时 standalone 要在输出中声明，"no"时不必在输出中声明。默认是"no"。 Netscape 6 不支持
doctype-public	string	可选。设置输出的 public DOCTYPE 值。
doctype-system	string	可选。设置输出的 system DOCTYPE。
cdata-section-elements	namelist	可选。用空格来分割内容为字符数据类型的元素。（ps:有疑问）
indent	yes no	可选。缩进选项。"yes"时要根据结构缩进，"no"时不缩进。默认是"yes" Netscape 6 不支持
media-type	string	可选。定义输出的 MIME 类型。默认是"text/xml" Netscape 6 不支持

例子 1

下面这段代码以 XML 格式输出，xml 版本是 1.0，编码是"ISO-8859-1"，并且为了易读文档要缩进。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" version="1.0"
encoding="iso-8859-1" indent="yes"/>.....</xsl:stylesheet>
```

例子 2

下面这段代码以 HTML 格式输出，HTML 版本是 4.0，编码是"ISO-8859-1"，并且为了易读文档要缩进。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:output method="html" version="4.0"
encoding="iso-8859-1" indent="yes"/>.....</xsl:stylesheet>
```

<xsl:param> 元素

定义与用法

<xsl:param>元素被用来声明一个局部或者全局的参数。

说明：如果声明的是一个高级（top-level）的元素那么参数是全局的，如果在一个样规中声明那么是局部的。

语法

```
<xsl:param
name="name"
select="expression"><!-- Content:template --></xsl:param>
```

属性

属性	值	说明
name	name	必须。指定参数的名称。
select	expression	可选。指定一个 XPath 表达式作为参数的默认值。

例子 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:variable name="xx">
  <html>
  <body>
  <xsl:call-template name="show_title">
    <xsl:with-param name="title" />
  </xsl:call-template>
  </body>
  </html>
</xsl:variable>
<xsl:template name="show_title" match="/">
  <xsl:param name="title" />
  <xsl:for-each select="catalog/cd">
    <p>Title: <xsl:value-of select="$title" /></p>
  </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

<xsl:preserve-space> 与 <xsl:strip-space> 元素

定义与用法

<xsl:preserve-space>元素用来定义必须保留空白空间的元素。

<xsl:strip-space>元素用来定义必须移除空白空间的元素。

说明：因为保留空白空间是默认设置，所以只有当<xsl:strip-space>元素被使用时才有必要使用<xsl:preserve-space>元素。

说明：<xsl:preserve-space>元素和<xsl:strip-space>元素是高级（top-level）元素。

语法

```
<xsl:preserve-space elements="list-of-element-names"/>

<xsl:strip-space elements="list-of-element-names"/>
```

属性

属性	值	说明
elements	list-of-element-names	必须。对列表中的元素加上/除去白色空间。 **说明：**列表可以包含“*”和“prefix:*”这样所有的元素或者一个特定的命名空间所有的元素能够加入。

例子 1

下面的这个例子，title 和 artist 元素保留白色空间，country, company, price, 和 year 元素移除白色空间：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:strip-space elements="country company price year" />
<xsl:preserve-space elements="title artist" />
<xsl:template match="/">
  <html>
  <body>
  <xsl:for-each select="catalog/cd">
```

```
      <p>
      <xsl:value-of select="title" /><br />
      <xsl:value-of select="artist" /><br />
      <xsl:value-of select="country" /><br />
      <xsl:value-of select="company" /><br />
      <xsl:value-of select="price" /><br />
      <xsl:value-of select="year" />
      </p>
   </xsl:for-each>
   </body>
   </html>
</xsl:template>

</xsl:stylesheet>
```

<xsl:processing-instruction> 元素

定义与用法

<xsl:processing-instruction>元素向输出写一条执行指令。

语法

```
<xsl:processing-instruction name="process-name">
<!-- Content:template -->
</xsl:processing-instruction>
```

属性

属性	值	说明
name	process-name	必须。指定执行指令的名称。

例子 1

下面的代码：

```
<xsl:processing-instruction name="xml-stylesheet">
href="style.css" type="text/css"
</xsl:processing-instruction>
```

生成如下的标记：

<?xml-stylesheet href="style.css" type="text/css"?>

<xsl:sort> 元素

定义与用法

<xsl:sort>元素被用来对输出排序。

说明：<xsl:sort>通常与<xsl:for-each> 或者 <xsl:apply-templates>一起使用。

语法
<xsl:sort
select="expression"
lang="language-code"
data-type="text|number|qname"
order="ascending|descending"
case-order="upper-first|lower-first"/>

属性

属性	值	说明
select	XPath-expression	可选。指定根据哪一个 node/node-set 为关键字来排序。
lang	language-code	可选。指定用哪一种语言来排序。
data-type	text number qname	可选。指定数据用来排序的数据类型。默认是“text”
order	ascending descending	可选。指定排序方式。默认是升序。
case-order	upper-first lower-first	可选。指定是按大写字母先排序还是按小写字母先排序。

例子 1
The example below will sort the output by artist:
下面例子的输出会以 artist 为关键字排序。
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
  <html>

```
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <xsl:sort select="artist"/>
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

<xsl:stylesheet> 与 <xsl:transform> 元素

定义与用法

<xsl:stylesheet>与<xsl:transform>元素是两个意思完全一样的元素。都被用来定义样式表的根元素。

语法

```
<xsl:stylesheet id="name" version="version" extension-element-prefixes="list"
exclude-result-prefixes="list">
<!-- Content:(<xsl:import>*,top-level-elements) -->
</xsl:stylesheet>
```

```
<xsl:transform id="name" version="version" extension-element-prefixes="list"
exclude-result-prefixes="list">
<!-- Content:(<xsl:import>*,top-level-elements) -->
</xsl:transform>
```

属性

属性	值	说明
version	version	必须。指定样式表的 XSLT 版本
extension-element-prefixes	list	可选。 扩展元素的命名空间前缀（以空格分割） Netscape 6 不支持该属性
exclude-result-prefixes	list	可选。 不因该在输出中出现的扩展元素的命名空间前缀（以空格分割）
id	name	可选。样式表的唯一 id Netscape 6 不支持该属性

例子 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">.....</xsl:stylesheet>
```

例子 2

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">.....</xsl:transform>
```

<xsl:template> 元素

定义与用法

The <xsl:template> element contains rules to apply when a specified node is matched.
<xsl:template>元素包含一些对匹配节点有用的规则。

match 属性用来匹配 XML 节点。也可以匹配全部 XML 文档（比如 match="/"）
说明: <xsl:template>是一个高级（top-level）元素。

语法

```
<xsl:template name="name" match="pattern" mode="mode" priority="number">
   <!-- Content:(<xsl:param>*,template) -->
</xsl:template>
```

Attributes

属性	值	说明
name	name	可选。指定该 template 的名称。 **说明：** 如果这个属性被省略了，一定会有 match 属性。
match	pattern	可选。template 匹配的模式 **说明：**如果这个属性被省略了，一定会有 name 属性
mode	mode	可选。 指定这个 template 的模式
priority	number	可选。 用一个数字指出了 template 的数字的优先级

例子

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>
<xsl:template match="cd">
  <p>
  <xsl:apply-templates select="title"/>
  <xsl:apply-templates select="artist"/>
  </p>
</xsl:template>
<xsl:template match="title">
  Title: <span style="color:#ff0000">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>
<xsl:template match="artist">
  Artist: <span style="color:#00ff00">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>
</xsl:stylesheet>
```

<xsl:text> 元素

定义与用法
<xsl:text> 元素 用来向输出写一段文字。

提示： 这个元素可以包含文字，实体引用和任何非 xml 元素的数据。

语法

```
<xsl:text disable-output-escaping="yes|no">
    <!-- Content:#PCDATA -->
</xsl:text>
```

属性

属性	值	说明
disable-output-escaping	yes no	可选。当为"yes"时，特殊的字符（比如"<"）照原样输出，当为"no"时，输出为"<"。默认是"no"。 这个属性在 Netscape 6 中不被支持

例子 1
展示每一个 CD 的 title。
1、 在每一个 CD-title 后插入“,”（条件：不是最后一个）
2、 如果是倒数第二个插入“, and”
3、 如果是最后一个插入“!”
显示每一个 CD 的标题。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <p>Titles:
    <xsl:for-each select="catalog/cd">
      <xsl:value-of select="title"/>
      <xsl:if test="position() < last()-1">
        <xsl:text>, </xsl:text>
```

```
        </xsl:if>
        <xsl:if test="position()=last()-1">
          <xsl:text>, and </xsl:text>
        </xsl:if>
        <xsl:if test="position()=last()">
          <xsl:text>!</xsl:text>
        </xsl:if>
      </xsl:for-each>
      </p>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

<xsl:value-of> 元素

定义与用法

<xsl:value-of>元素取出选择的节点的值。

<xsl:value-of>元素可以被用来取出 XML 元素的值，并输出。

说明：value 使用 XPath 表达式取值，使用像文件系统那样的方法用(/)选择子文件夹。

语法

```
<xsl:value-of select="expression" disable-output-escaping="yes|no"/>
```

属性

属性	值	说明
select	expression	必须。一个 XPath 表达式用来指定从哪一个节点/属性来取值。
disable-output-escaping	yes no	可选。当为"yes"时，特殊的字符（比如"<"）照原样输出，当为"no"时，输出为"<"。默认是"no"。

例子 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
```

```
 <html>
 <body>
   <h2>My CD Collection</h2>
   <table border="1">
     <tr bgcolor="#9acd32">
       <th>Title</th>
       <th>Artist</th>
     </tr>
     <tr>
      <td><xsl:value-of select="catalog/cd/title"/></td>
      <td><xsl:value-of select="catalog/cd/artist"/></td>
     </tr>
   </table>
 </body>
 </html>
</xsl:template></xsl:stylesheet>
```

例子 2

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template></xsl:stylesheet>
```

<xsl:variable> 元素

定义与用法

<xsl:variable>元素被用来声明一个局部或者全局的变量。

说明：当声明为高级（top-level）元素时，变量是全局的，当在一个 template 中声明时，作为局部变量。

说明：一旦您设置了一个变量的值，您就不能修改这个值！

提示：您可以通过增加<xsl:variable>的内容或者用 select 属性来设置变量值！

语法

```
<xsl:variable name="name" select="expression">
  <!-- Content:template -->
</xsl:variable>
```

属性

属性	值	说明
name	name	必须。指定变量名称
select	expression	可选。指定变量的值

例子 1

如果 select 属性被设置了，那么<xsl:variable>元素就不可以包含任何内容。如果 select 属性包含字符串，则在设置时必须用引号。

下面的两行示例代码分别给变量"color"赋值为"red"。

```
<xsl:variable name="color" select="'red'" />

<xsl:variable name="color" select="'red'" />
```

例子 2

如果<xsl:variable>元素仅有 name 属性，元素间没有内容，那么就定义了一个空字符串变量。

```
<xsl:variable name="j" />
```

例子 3

下面的这个例子用<xsl:variable>元素将

```
<tr>
    <th>title</th>
    <th>year</th>
```

</tr>
作为变量传递给输出：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:variable name="header">
        <tr>
            <th>title</th>
            <th>year</th>
        </tr>
    </xsl:variable>
    <xsl:template match="/">
        <html>
            <body>
                <table>
                    <xsl:copy-of select="$header"/>
                    <xsl:text>
                    year    = 1982
                    </xsl:text>
                    <xsl:for-each select="catalog/cd">
                        <xsl:if test="year    = 1982">
                            <tr>
                                <td>
                                    <xsl:value-of select="title"/>
                                </td>
                                <td>
                                    <xsl:value-of select="year"/>
                                </td>
                            </tr>
                        </xsl:if>
                    </xsl:for-each>
                </table>
                <br/>
                <table>
                    <xsl:copy-of select="$header"/>
                    <xsl:text>
                    year    = 1985
                    </xsl:text>
                    <xsl:for-each select="catalog/cd">
                        <xsl:if test="year = 1985">
                            <tr>
                                <td>
                                    <xsl:value-of select="title"/>
                                </td>
                                <td>
```

```
                                        <xsl:value-of select="year"/>
                                    </td>
                                </tr>
                            </xsl:if>
                        </xsl:for-each>
                    </table>
                </body>
            </html>
        </xsl:template>
</xsl:stylesheet>
```

<xsl:when> 元素

定义和用法

<xsl:when>元素为<xsl:choose>元素指定一个动作。当符合<xsl:when>元素的条件时，动作就被激活。

说明：<xsl:when>元素与<xsl:choose> 和 <xsl:otherwise>元素一同使用，来表达多条件选择。

语法

```
<xsl:when test="boolean-expression">
<!-- Content: template -->
</xsl:when>
```

属性

属性	值	说明
test	boolean-expression	必须。指定执行条件。

例子 1

下面的这段代码将把 cd /price 高于 10 artist 列的背景色设为粉红色。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
```

```
            <th>Title</th>
            <th>Artist</th>
          </tr>
          <xsl:for-each select="catalog/cd">
          <tr>
            <td><xsl:value-of select="title"/></td>
            <xsl:choose>
              <xsl:when test="price&gt;'10'">
                <td bgcolor="#ff00ff">
                <xsl:value-of select="artist"/></td>
              </xsl:when>
              <xsl:otherwise>
                <td><xsl:value-of select="artist"/></td>
              </xsl:otherwise>
            </xsl:choose>
          </tr>
          </xsl:for-each>
        </table>
    </body>
    </html>
</xsl:template></xsl:stylesheet>
```

例子 2

声明一个名为"color"的变量。把当前元素 color 属性的值赋给"color"变量。如果当前元素没有 color 属性，则"color"变量的值将为"green"：

```
<xsl:variable name="color">
    <xsl:choose>
      <xsl:when test="@color">
        <xsl:value-of select="@color"/>
      </xsl:when>
      <xsl:otherwise>green</xsl:otherwise>
    </xsl:choose>
</xsl:variable>
```

<xsl:with-param> 元素

定义与用法

<xsl:with-param>元素向模板传递的参数的值。

说明：<xsl:with-param>元素的 name 属性的值必须与一个<xsl:param>元素的 name 相匹配（否则<xsl:with-param>元素将被忽略）。

说明：<xsl:with-param>元素在<xsl:apply-templates>和<xsl:call-template>之内使用。

提示：您可以通过两种方式向<xsl:with-param>元素赋值

1、 用 select 属性。

2、 在<xsl:with-param name="A">与</xsl:with-param>元素中写。

语法

```
<xsl:with-param name="name" select="expression">
<!-- Content:template -->
</xsl:with-param>
```

属性

属性	值	说明
name	name	必须。指定参数的名字
select	expression	可选。 An XPath expression that defines the value of the parameter 一个 XPath 表达式定义了参数的值

例子 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="text"/>
    <xsl:template match="/">
        <xsl:call-template name="show_title">
            <xsl:with-param name="title" select="catalog/cd/title"/>
        </xsl:call-template>
    </xsl:template>

    <xsl:template name="show_title">
        <xsl:param name="title"/>
        <xsl:for-each select="catalog/cd">
            <p>
            Title: <xsl:value-of select="$title"/>
            </p>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

例子 2

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:output method="text"/>
    <xsl:template match="/">
        <xsl:call-template name="print">
            <xsl:with-param name="A">11</xsl:with-param>
            <xsl:with-param name="B">33</xsl:with-param>
        </xsl:call-template>
        <xsl:call-template name="print">
            <xsl:with-param name="A">55</xsl:with-param>
        </xsl:call-template>
    </xsl:template>
    <xsl:template name="print">
        <xsl:param name="A"/>
        <xsl:param name="B">111</xsl:param>
        <xsl:text>

        </xsl:text>
        <xsl:value-of select="$A"/>
        <xsl:text> + </xsl:text>
        <xsl:value-of select="$B"/>
        <xsl:text> = </xsl:text>
        <xsl:value-of select="$A+$B"/>
    </xsl:template>
</xsl:stylesheet>
```

这个例子的输出结果为

11 + 33 = 44 55 + 111 = 166

XSLT 函数

current() 函数

定义与用法

current()函数返回一个仅包含当前节点的节点集。通常当前节点和上下文节点是一样的。

<xsl:value-of select="current()"/>与<xsl:value-of select="."/>是一样的。

然而，有一点不同。看下面的 XPath 表达式："catalog/cd"。这个表达式选择<catalog>作为当前节点，然后选择<catalog>的子节点<cd>。这意味着在赋值的每一步，"."都有不同的意义。

下面这行代码：
将会处理所有 title 属性的值与当前节点的 ref 属性的值相等的所有 cd 元素。
<xsl:apply-templates select="//cd[@title=current()/@ref]"/>

这与下面有区别
下面这段代码将会处理所有 title 属性的值与 ref 属性的值相等的所有 cd 元素。
<xsl:apply-templates select="//cd[@title=./@ref]"/>

语法
node-set current()

例子 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
  <html>
  <body>
  <xsl:for-each select="catalog/cd/artist">
    Current node: <xsl:value-of select="current()"/>
    <br />
  </xsl:for-each>
  </body>
  </html>
</xsl:template>
```

</xsl:stylesheet>

document() 函数

定义与用法
document()函数用来访问外部的 XML 文档。外部 XML 文档必须是正确可解析的。

<xsl:value-of select="document('celsius.xml')/celsius/result[@value=$value]"/>

语法
node-set document(object,node-set?)

参数

参数	说明
object	必须。定义 XML 文档的 URI。
node-set	可选。用来处理相关的 URI。

例子 1
Document.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="document.xsl" ?>
<groups>
	<groupRef>hrGroup.xml</groupRef>
	<groupRef>myGroup.xml</groupRef>
</groups>
```

document.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
	<xsl:template match="/">
		<groups>
			<xsl:apply-templates select="/groups/groupRef"/>
		</groups>
	</xsl:template>
	<xsl:template match="groups/groupRef">
		<xsl:copy-of select="document(.)//group"/>
```

```
    </xsl:template>
</xsl:stylesheet>
```

hrGroup.xml

```
<?xml version='1.0'?>
<group name="hr">
   <leader>mo</leader>
   <member>bo</member>
   <member>ko</member>
   <member>lo</member>
</group>
```

myGroup.xml

```
<?xml version='1.0'?>
<group name="my">
   <leader>john</leader>
   <member>jane</member>
   <member>jon</member>
   <member>jan</member>
</group>
```

例子 2

下面这个例子，显示如何使用两个参数。

第二个参数必须是一个节点集，作为第一个参数的基准 URI，当没有第二个参数时，第一个参数的基准 URI 就是 XSL 的 URI，如下，把 a.xml 放到 subdir 目录下，而另外两个在../，如果 document('a.xml', .)的第二个参数不写，将以../作为 a.xml 的目录，就会报错，您可以实验一下。

document2.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

    <xsl:template match="/">
    <root>
        <xsl:comment>One Argument </xsl:comment>
        <xsl:for-each select="document('b.xml')//a">
            <xsl:copy-of select="."/>
        </xsl:for-each>

        <xsl:comment>Two Argument </xsl:comment>
        <xsl:for-each select="document('a.xml', .)//a">
            <xsl:copy-of select="."/>
        </xsl:for-each>
    </root>
```

```
    </xsl:template>
</xsl:stylesheet>
```

Subdir/a.xml

```
<?xml-stylesheet type="text/xsl" href="../document2.xsl" ?>
<doc>
<a> I </a>
<a> II </a>
<a> III </a>
</doc>
```

b.xml

```
<doc>
  <a> one </a>
  <a> two </a>
  <a> three </a>
</doc>
```

element-available() 函数

定义与用法

element-available()函数检查 XSLT 处理器是否能处理指定的元素。

这个函数只能用来检查在模板里出现的元素，这些元素是：
xsl:apply-imports
xsl:apply-templates
xsl:attributes
xsl:call-template
xsl:choose
xsl:comment
xsl:copy
xsl:copy-of
xsl:element
xsl:fallback
xsl:for-each
xsl:if
xsl:message
xsl:number
xsl:processing instruction
xsl:text
xsl:value-of

xsl:variable

语法
boolean element-available(string)

参数

参数	说明
string	必须。指定要检查元素。

例子 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
<html>
<body>
<xsl:choose>
<xsl:when test="element-available('xsl:comment')">
<p>xsl:comment is supported.</p>
</xsl:when>
<xsl:otherwise>
<p>xsl:comment is not supported.</p>
</xsl:otherwise>
</xsl:choose>
<xsl:choose>
<xsl:when test="element-available('xsl:delete')">
<p>xsl:delete is supported.</p>
</xsl:when>
<xsl:otherwise>
<p>xsl:delete is not supported.</p>
</xsl:otherwise>
</xsl:choose>
</body>
</html>
</xsl:template></xsl:stylesheet>
```

format-number() 函数

定义与用法

format-number()函数用来格式化一个数字字符串。

语法
string format-number(number,format,[decimalformat])

参数

参数	说明
number	Required. Specifies the number to be formatted 必须。指定要格式化的数字
format	必须。指定格式化的模式。 • # (指示一个数字。 比如： ####) • 0 (Denotes leading and following zeros. Example: 0000.00) • . (指定小数点的位置。比如: ###.##) • , (The group separator for thousands. Example: ###,###.##) • % (Displays the number as a percentage. Example: ##%) • ; (Pattern separator. The first pattern will be used for positive numbers and the second for negative numbers)
decimalformat	可选。用来指定使用的<xsl:decimal-format>元素的名称。

例子 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
<html>
<body>
<xsl:value-of select='format-number(500100, "#.00")' />
<br />
<xsl:value-of select='format-number(500100, "#.0")' />
<br />
<xsl:value-of select='format-number(500100, "###,###.00")' />
<br />
<xsl:value-of select='format-number(0.23456, "##%")' />
<br />
<xsl:value-of select='format-number(500100, "#######")' />
</body>
</html>
</xsl:template>
```

</xsl:stylesheet>

function-available() 函数

定义与用法

function-available()函数检查指定的函数在 XSLT 处理器中是否支持。
You may test the XSLT functions and the inherited XPath functions.
您可以检查 XSLT 函数和 XPath 函数。

语法
boolean function-available(string)

参数

参数	说明
string	必须。指定要检查的函数。

例子 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<xsl:choose>
<xsl:when test="function-available('sum')">
<p>sum() is supported.</p>
</xsl:when>
<xsl:otherwise>
<p>sum() is not supported.</p>
</xsl:otherwise>
</xsl:choose>
<xsl:choose>
<xsl:when test="function-available('current')">
<p>current() is supported.</p>
</xsl:when>
```

<xsl:otherwise>
current() is not supported.
</xsl:otherwise>
</xsl:choose>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

generate-id() 函数

定义与用法

generate-id()函数返回一个唯一标示指定节点的字符串。

如果指定的节点集是空的，将返回一个空字符串。如果您忽略了节点集的参数，默认是当前节点。

语法
string generate-id(node-set?)

参数

参数	说明
node-set	可选。指定要生成唯一 id 的节点集

例子 1
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/">
<html>
<body>
<h3>Artists:</h3>

<xsl:for-each select="catalog/cd">

<xsl:value-of select="artist" />

```
</li>
</xsl:for-each>
</ul>
<hr />
<xsl:for-each select="catalog/cd">
Artist: <a name="{generate-id(artist)}">
<xsl:value-of select="artist" /></a>
<br />
Title: <xsl:value-of select="title" />
<br />
Price: <xsl:value-of select="price" />
<hr />
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

key() 函数

定义与用法

key()函数使用<xsl:key>元素指定的索引返回文档中的一个节点集。

语法

node-set key(string, object)

参数

参数	说明
string	必须。指定 xsl:key 元素的名称。
object	必须。要查找的字符串。

例子 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:key name="cdlist" match="cd"
use="title" />
```

```
<xsl:template match="/">
<html>
<body>
<xsl:for-each select="key('cdlist', 'Empire Burlesque')">
  <p>
  Title: <xsl:value-of select="title" />
  <br />
  Artist: <xsl:value-of select="artist" />
  <br />
  Price: <xsl:value-of select="price" />
  </p>
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

system-property() 函数

定义与用法

The system-property() function returns the value of the system property specified by the name.

system-property()函数返回指定名称的系统属性。

System properties in the XSLT namespace:

在 XSLT 名称空间里的系统属性：

xsl:version - 指出 XSLT 的版本

xsl:vendor - 指出 XSLT 处理器(比如 James Clark)

xsl:vendor-url - 指出处理器的 URL(比如 http://www.jclark.com/)

（注：当用 XT 解析时 xsl:vendor 是 James Clark，xsl:vendor-url 是 http://www.jclark.com/，用 IE 就为 Microsoft,http://www.microsoft.com）

语法

object system-property(string)

参数

参数	说明
string	必须。指定返回的系统属性

例子 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<p>
Version:
<xsl:value-of select="system-property('xsl:version')" />
<br />
Vendor:
<xsl:value-of select="system-property('xsl:vendor')" />
<br />
Vendor URL:
<xsl:value-of select="system-property('xsl:vendor-url')" />
</p>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

unparsed-entity-uri() 函数

定义与用法

unparsed-entity-uri()函数返回未解析的实体的 URI。实体名称必须与传递的参数匹配。如果没有这样的实体，那么返回空串。

如果 DTD 包含下面的声明：
<!ENTITY pic SYSTEM "http://www.w3schools.com/picture.jpg" NDATA JPEG>

下面的表达式
unparsed-entity-uri('pic')

将会返回"picture.jpg"的 URI。

语法
string unparsed-entity-uri(string)

参数

参数	说明
string	必须。指定未解析实体

例子 1

XML 文件

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="sample.xsl"?>
<!DOCTYPE catalog [
	<!ELEMENT catalog ANY>
	<!ELEMENT book ANY>
	<!ELEMENT title ANY>
	<!NOTATION JPEG SYSTEM "urn:foo">
	<!ENTITY pic SYSTEM "http://www.w3schools.com/picture.jpg" NDATA JPEG>
]>
<catalog>
	<book>
		<title>XML Developer's Guide</title>
	</book>
	<book>
		<title>Midnight Rain</title>
	</book>
</catalog>
```

XSL 文件

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:output method="html"/>
	<xsl:template match="/">
		<html>
			<body>
				<h3>unparsed-entity-uri()</h3>
				<ul>
					<li>
						<b>unparsed-entity-uri('pic')</b> =
				<xsl:value-of select="unparsed-entity-uri('pic')"/>
					</li>
				</ul>
			</body>
		</html>
	</xsl:template>
```

</xsl:stylesheet>

运算符和特殊字符

!= 不等于
"" (literal) 文字
'' (literal) 文字
() (grouping) 分组
* (all nodes) 所有节点
* (multiplication) 通配符
+ 加
- 减
- (unary minus)
. (self axis short form) 当前元素
.. (parent axis short form) 父元素
/ (step separator) 选择子元素
// (descendant-or-self axis short form) 选择子元素，不论多深
:: (axis specifier) （不知道怎么翻译合适，没用过）
< 小于
<= 小于等于
= 等于
> 大于
>= 大于等于
@ (attribute axis short form) 选择属性
@* (all attributes) 选择所有属性
[] (predicate) 断言，指定过滤样式
And 与
axis nodetest predicate (step)
div The div operator performs floating-point division according to IEEE 754.（原文放上来，不知道怎么翻译合适，没用过）
func() (function call) 调用函数
mod 取余
name (node test)
or 或
| (union) 集合

附录

如果没有特殊说明，将使用下面的 XML 文件

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!—在这里引入您的 XSL 文件 -->
<?xml-stylesheet type="text/xsl" href="cdcatalog_element.xsl"?>
<catalog>
    <cd>
```

```
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
</cd>
<cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
    <company>CBS Records</company>
    <price>9.90</price>
    <year>1988</year>
</cd>
<cd>
    <title>Greatest Hits</title>
    <artist>Dolly Parton</artist>
    <country>USA</country>
    <company>RCA</company>
    <price>9.90</price>
    <year>1982</year>
</cd>
<cd>
    <title>Still got the blues</title>
    <artist>Gary Moore</artist>
    <country>UK</country>
    <company>Virgin records</company>
    <price>10.20</price>
    <year>1990</year>
</cd>
<cd>
    <title>Eros</title>
    <artist>Eros Ramazzotti</artist>
    <country>EU</country>
    <company>BMG</company>
    <price>9.90</price>
    <year>1997</year>
</cd>
<cd>
    <title>One night only</title>
    <artist>Bee Gees</artist>
    <country>UK</country>
    <company>Polydor</company>
```

```
        <price>10.90</price>
        <year>1998</year>
</cd>
<cd>
        <title>Sylvias Mother</title>
        <artist>Dr.Hook</artist>
        <country>UK</country>
        <company>CBS</company>
        <price>8.10</price>
        <year>1973</year>
</cd>
<cd>
        <title>Maggie May</title>
        <artist>Rod Stewart</artist>
        <country>UK</country>
        <company>Pickwick</company>
        <price>8.50</price>
        <year>1990</year>
</cd>
<cd>
        <title>Romanza</title>
        <artist>Andrea Bocelli</artist>
        <country>EU</country>
        <company>Polydor</company>
        <price>10.80</price>
        <year>1996</year>
</cd>
<cd>
        <title>When a man loves a woman</title>
        <artist>Percy Sledge</artist>
        <country>USA</country>
        <company>Atlantic</company>
        <price>8.70</price>
        <year>1987</year>
</cd>
<cd>
        <title>Black angel</title>
        <artist>Savage Rose</artist>
        <country>EU</country>
        <company>Mega</company>
        <price>10.90</price>
        <year>1995</year>
</cd>
<cd>
```

```
    <title>1999 Grammy Nominees</title>
    <artist>Many</artist>
    <country>USA</country>
    <company>Grammy</company>
    <price>10.20</price>
    <year>1999</year>
</cd>
<cd>
    <title>For the good times</title>
    <artist>Kenny Rogers</artist>
    <country>UK</country>
    <company>Mucik Master</company>
    <price>8.70</price>
    <year>1995</year>
</cd>
<cd>
    <title>Big Willie style</title>
    <artist>Will Smith</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>9.90</price>
    <year>1997</year>
</cd>
<cd>
    <title>Tupelo Honey</title>
    <artist>Van Morrison</artist>
    <country>UK</country>
    <company>Polydor</company>
    <price>8.20</price>
    <year>1971</year>
</cd>
<cd>
    <title>Soulsville</title>
    <artist>Jorn Hoel</artist>
    <country>Norway</country>
    <company>WEA</company>
    <price>7.90</price>
    <year>1996</year>
</cd>
<cd>
    <title>The very best of</title>
    <artist>Cat Stevens</artist>
    <country>UK</country>
    <company>Island</company>
```

```
        <price>8.90</price>
        <year>1990</year>
</cd>
<cd>
        <title>Stop</title>
        <artist>Sam Brown</artist>
        <country>UK</country>
        <company>A and M</company>
        <price>8.90</price>
        <year>1988</year>
</cd>
<cd>
        <title>Bridge of Spies</title>
        <artist>T`Pau</artist>
        <country>UK</country>
        <company>Siren</company>
        <price>7.90</price>
        <year>1987</year>
</cd>
<cd>
        <title>Private Dancer</title>
        <artist>Tina Turner</artist>
        <country>UK</country>
        <company>Capitol</company>
        <price>8.90</price>
        <year>1983</year>
</cd>
<cd>
        <title>Midt om natten</title>
        <artist>Kim Larsen</artist>
        <country>EU</country>
        <company>Medley</company>
        <price>7.80</price>
        <year>1983</year>
</cd>
<cd>
        <title>Pavarotti Gala Concert</title>
        <artist>Luciano Pavarotti</artist>
        <country>UK</country>
        <company>DECCA</company>
        <price>9.90</price>
        <year>1991</year>
</cd>
<cd>
```

```
        <title>The dock of the bay</title>
        <artist>Otis Redding</artist>
        <country>USA</country>
        <company>Atlantic</company>
        <price>7.90</price>
        <year>1987</year>
    </cd>
    <cd>
        <title>Picture book</title>
        <artist>Simply Red</artist>
        <country>EU</country>
        <company>Elektra</company>
        <price>7.20</price>
        <year>1985</year>
    </cd>
    <cd>
        <title>Red</title>
        <artist>The Communards</artist>
        <country>UK</country>
        <company>London</company>
        <price>7.80</price>
        <year>1987</year>
    </cd>
    <cd>
        <title>Unchain my heart</title>
        <artist>Joe Cocker</artist>
        <country>USA</country>
        <company>EMI</company>
        <price>8.20</price>
        <year>1987</year>
    </cd>
</catalog> xiexie
```