1. **标头**

XSL 样式表本身也是一个 XML 文档，因此它总是由 XML 声明起始：

<?xml version="1.0" encoding="ISO-8859-1"?>

XSL 样式表由一个或多套被称为模板（template）的规则组成。

每个模板含有当某个指定的节点被匹配时所应用的规则。

1. **样式表声明**

如需访问 XSLT 的元素、属性以及特性，我们必须在文档顶端声明 XSLT 命名空间。

|  |
| --- |
| <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  或者  <xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> |

xmlns:xsl="http://www.w3.org/1999/XSL/Transform" 指向了官方的 W3C XSLT 命名空间。如果您使用此命名空间，就必须包含属性 version="1.0"。

1. **把 XSL 样式表链接到 XML 文档**

<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>

1. **<xsl:template>**

<xsl:template> 元素用于构建模板。

<xsl:template match="/"> match 属性也可用来为整个文档定义模板。match 属性的值是 XPath 表达式（举例，match="/" 定义整个文档）。

1. **<xsl:value-of>**

提取某个选定节点的值，并把值添加到转换的输出流中

|  |
| --- |
| <tr>  <td><xsl:value-of select="catalog/cd/title"/></td>  <td><xsl:value-of select="catalog/cd/artist"/></td>  </tr> |

select 属性的值是一个 XPath 表达式。

1. **<xsl:for-each>**

用于选取指定的节点集中的每个 XML 元素。

|  |
| --- |
| <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> |

## 过滤运算符:

|  |
| --- |
| = (等于) !=(不等于) &lt;(小于) &gt;(大于) |

|  |
| --- |
| <xsl:for-each select="catalog/cd[artist='Bob Dylan']">  <tr>  <td><xsl:value-of select="title"/></td>  <td><xsl:value-of select="artist"/></td>  </tr>  </xsl:for-each> |

1. **<xsl:sort>**

对结果进行排序。

如需对结果进行排序，只要简单地在 XSL 文件中的 <xsl:for-each> 元素内部添加一个 <xsl:sort> 元素

|  |
| --- |
| <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> |

1. **<xsl:if>**

放置针对 XML 文件内容的条件测试。

|  |
| --- |
| <xsl:for-each select="catalog/cd">  <xsl:if test="price &gt; 10">  <tr>  <td><xsl:value-of select="title"/></td>  <td><xsl:value-of select="artist"/></td>  </tr>  </xsl:if>  </xsl:for-each> |

1. **<xsl:choose>**

<xsl:choose> 元素用于结合 <xsl:when> 和 <xsl:otherwise> 来表达多重条件测试。

|  |
| --- |
| <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> |

1. **<xsl:attribute>**

Xml文档

|  |
| --- |
| <?xml version="1.0" encoding="ISO-8859-1"?>  <tool>  <field id="prodName">  <value>HAMMER HG2606</value>  </field>  <field id="prodNo">  <value>32456240</value>  </field>  <field id="price">  <value>$30.00</value>  </field>  </tool> |

XSLT文档

|  |
| --- |
| <?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>  <form method="post" action="edittool.asp">  <h2>Tool Information (edit):</h2>  <table border="0">  <xsl:for-each select="tool/field">  <tr>  <td>  <xsl:value-of select="@id"/>  </td>  <td>  <input type="text">  <xsl:attribute name="id">  <xsl:value-of select="@id" />  </xsl:attribute>  <xsl:attribute name="name">  <xsl:value-of select="@id" />  </xsl:attribute>  <xsl:attribute name="value">  <xsl:value-of select="value" />  </xsl:attribute>  </input>  </td>  </tr>  </xsl:for-each>  </table>  <br />  <input type="submit" id="btn\_sub" name="btn\_sub" value="Submit" />  <input type="reset" id="btn\_res" name="btn\_res" value="Reset" />  </form>  </body>  </html>  </xsl:template>  </xsl:stylesheet> |

上面这个 XSL 文件会循环遍历 XML 文件中的元素，并为每个 XML "field" 元素创建一个输入域。field 元素的 id 属性的值被添加到每个 HTML 输入域的 id 和 name 属性。"value" 元素的值被添加到每个 HTML 输入域的 "value" 属性。结果是，可以得到一个包含 XML 文件中值的可编辑的 HTML 表单。

1. **<xsl:apply-templates>**

把一个模板应用于当前的元素或者当前元素的子节点。

假如我们向 <xsl:apply-templates> 元素添加一个 select 属性，此元素就会仅仅处理与属性值匹配的子元素。我们可以使用 select 属性来规定子节点被处理的顺序。

|  |
| --- |
| <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> |

1. **将XML 文档转换为 XHTML**

* **使用XSLT转换**

把 XSL 样式表链接到 XML 文档;

* **在客户端/浏览器利用Javascript将XML转换成XHTML**

|  |
| --- |
| <html>  <body>  <script type="text/javascript">  // Load XML  var xml = new ActiveXObject("Microsoft.XMLDOM")  xml.async = false  xml.load("cdcatalog.xml")  // Load XSL  var xsl = new ActiveXObject("Microsoft.XMLDOM")  xsl.async = false  xsl.load("cdcatalog.xsl")  // Transform  document.write(xml.transformNode(xsl))  </script>  </body>  </html> |

这种方法的好处：XML 文件可使用多个不同的 XSL 样式表来进行转换;

* **在服务器将XML转换成XHTML**

|  |
| --- |
| <%  'Load XML  set xml = Server.CreateObject("Microsoft.XMLDOM")  xml.async = false  xml.load(Server.MapPath("cdcatalog.xml"))  'Load XSL  set xsl = Server.CreateObject("Microsoft.XMLDOM")  xsl.async = false  xsl.load(Server.MapPath("cdcatalog.xsl"))  'Transform file  Response.Write(xml.transformNode(xsl))  %> |