# 实验六 使用XSL将XML文档变换为HTML文档

**一、实验目的**

掌握XSLT语言的基本标签的使用；熟悉XML文档的树状结构以及与XSL文档树状结构的关系；了解XSL处理器对于XSL文档的处理方式。

**二、实验原理及相关知识**

* + - **XSL文档的声明**

声明根据W3C XSLT参考标准的XSL样式表的正确的途径是：

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

由于XSLT语言是遵循XML文档语法规则的，所以在XSL文档的最前面也必须加上XML文档的声明。

* + - **XSLT语言的基本元素**

XSL样式表是由一个或者更多的被称为“模板（templates）”的规则设置（set of rules）组成的。

示例1：Employees.xml文档（后面的所有XSL文档示例都以此XML文档为依据）

|  |
| --- |
| **<?xml version="1.0" encoding="utf-8"?>**  **<?xml-stylesheet type="text/xsl" href="所要引用的XSL文档的本地路径"?>**  <!--这是公司雇员的信息-->  <employees>  <employee sn="E-200402100001" >  <name>zhangsan</name>  <age>25</age>  <!--月薪小于等于2000元的雇员工资，以现金方式发放-->  <monthly\_pay mode="cash">  1200.00  </monthly\_pay>  </employee>  <employee sn="E-200402100006" >  <name>lisi</name>  <age>28</age>  <!--月薪小于等于2000元的雇员工资，以现金方式发放-->  <monthly\_pay mode="cash">  1600.00  </monthly\_pay>  </employee>  <employee sn="E-200503220001" >  <name>wangwu</name>  <age>30</age>  <!--月薪高于2000元的雇员工资，以信用卡转帐的方式发放-->  <monthly\_pay mode="credit\_card">  3500.00  </monthly\_pay>  </employee>  </employees> |

#### <xsl:template>元素

该元素用来定义对于某个XSL树状结构中的节点的转换模板。在该元素中包含一个match属性，使用标记匹配模式指定该模板所要应用的特定节点。

示例2：employees\_simple.xsl

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >  <xsl:template match="/">  <html>  <head>  <title>雇员月薪表</title>  </head>  <body>  <xsl:apply-templates/>  </body>  </html>  </xsl:template>  </xsl:stylesheet> |

显示结果如图4-1所示：



图1 例2显示结果

实际上，如果在XML文档中引用了一个XSL文档，但是在XSL文档中并没有定义任何文本的话，使用IE来打开该XML文档时，所显示出的内容会和上面所给出的XSL文档被应用时所显示的内容是相同的。这是因为XSLT中定义了几个默认模板规则，当引用了XSL文档，但是XSL文档中又没有显式的定义特定的模板规则话，XSLT处理器会自动调用默认模板规则。

示例3：（将上面的XSL文档中的模板规则删除即得以下内容）

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >  </xsl:stylesheet> |

显示结果如图2所示，可以看出这与示例2的变换结果相似。



图2 例3显示结果

#### <xsl:apply-templates>元素

该元素会把模板应用到当前元素或当前元素的子节点上。

如果用户向只处理与属性值相匹配的子元素的<xsl:apply-templates>中添加一个select属性。

示例4：employees\_apply\_templates\_select.xsl

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >  <xsl:template match="/">  <html>  <head>  <title>雇员月薪表</title>  </head>  <body>  <table border="1">  <xsl:apply-templates select="//monthly\_pay"/>  </table>  </body>  </html>  </xsl:template>  <xsl:template match="monthly\_pay">  <tr>  <td><xsl:value-of select="."/></td>  </tr>  </xsl:template>  </xsl:stylesheet> |

在上面的XSL文档中具有两个模板规则，分别是：<xsl:template match="/">和<xsl:template match="monthly\_pay"> 。在与”/”（XSL文档根节点）匹配的模板规则中，给出了一个<xsl:apply-templates select="//monthly\_pay"/> 用来说明在处理XSL文档根节点的所有子节点时，当遇到有名为：monthly\_pay的所有子节点，XSL处理器会在XSL文档中查找与之对应的模板规则，并将其应用到monthly\_pay节点。

显示结果如图3所示：

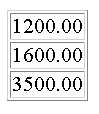


图3 例4显示结果

#### <xsl:value-of>元素

该元素可以用来选取XML元素节点的内容，并其作为文本节点添加到已被转换的输出流中去。

如示例4所示，其中的<xsl:value-of select="."/> 这是告诉XSL处理器在应用模板规则时，将选择当前节点的内容，把它作为文本节点输出到转换文档中。

#### <xsl:for-each>元素

该元素用来依次选择具有指定的节点设置（node-set）的每一个XML元素节点。

示例5：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >  <xsl:template match="/">  <html>  <head>  <title>雇员月薪表</title>  <style type="text/css">  table{background-color="lightgrey"}  </style>  </head>  <body>  <xsl:apply-templates />  </body>  </html>  </xsl:template>  <xsl:template match="employees">  <table border="1" width="400">  <tbody>  <tr>  <th>姓名</th><th>年龄</th><th>月薪</th>  </tr>  ***<xsl:for-each select="employee">***  ***<tr>***  ***<td><xsl:value-of select="name"/></td>***  ***<td><xsl:value-of select="age"/></td>***  ***<td><xsl:value-of select="monthly\_pay"/></td>***  ***</tr>***  </xsl:for-each>  </tbody>  </table>  </xsl:template>  </xsl:stylesheet> |

显示结果如图4所示：

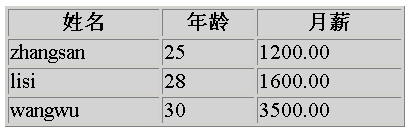


图4 例5显示结果

本例中使用<xsl:for-each select="employee"> 表示将选择所有名为employee标记的元素节点，并应用其后的模块规则。

#### <xsl:sort>元素

该元素用于对元素进行排序，该元素只能作为<xsl:apply-templates> 元素和的<xsl:for-each > 元素的子元素使用。当作为<xsl:for-each> 的子元素时，<xsl:sort> 必须第一个出现。

示例6：（作为<xsl:for-each>的子节点对名为name的元素节点进行升序排序）

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >  <xsl:template match="/">  <html>  <head>  <title>雇员月薪表</title>  <style type="text/css">  {background-color="lightgrey"}  </style>  </head>  <body>  <xsl:apply-templates />  </body>  </html>  </xsl:template>  <xsl:template match="employees">  <table border="1" width="400">  <tbody>  <tr>  <th>姓名</th><th>年龄</th><th>月薪</th>  </tr>  ***<xsl:for-each select="employee">***  ***<xsl:sort order="ascending" select="name"/>***  ***<tr>***  ***<td><xsl:value-of select="name"/></td>***  ***<td><xsl:value-of select="age"/></td>***  ***<td><xsl:value-of select="monthly\_pay"/></td>***  ***</tr>***  ***</xsl:for-each>***  </tbody>  </table>  </xsl:template>  </xsl:stylesheet> |

显示结果如图5所示：

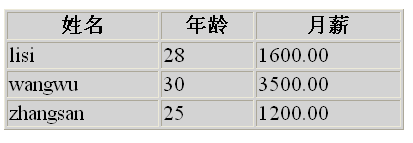


图5 例6显示结果

示例7：（作为<xsl:apply-templates>的子节点对名为name的元素节点进行升序排序）

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >  <xsl:template match="/">  <html>  <head>  <title>雇员月薪表</title>  </head>  <body>  <xsl:apply-templates />  </body>  </html>  </xsl:template>  <xsl:template match="employees">  <table border="1">  <tbody>  <tr>  <th>姓名</th><th>年龄</th><th>月薪</th>  </tr>  ***<xsl:apply-templates >***  ***<xsl:sort order="ascending" select="name"/>***  ***</xsl:apply-templates>***  </tbody>  </table>  </xsl:template>  <xsl:template match="employee">  <tr>  <td><xsl:value-of select="name"/></td>  <td><xsl:value-of select="age"/></td>  <td><xsl:value-of select="monthly\_pay"/></td>  </tr>  </xsl:template>  </xsl:stylesheet> |

显示结果如图6所示：



图6 例7显示结果

#### <xsl:if>元素

该元素用于对XML文件的内容设置一个条件语句。要添加一个条件语句，那必须将一个<xsl:if>元素放在XSL文件里的<xsl:for-each>里。

示例8：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  <xsl:template match="/">  <html>  <head>  <title>雇员月薪表</title>  <style type="text/css">  {background-color="lightgrey"}  </style>  </head>  <body>  <xsl:apply-templates/>  </body>  </html>  </xsl:template>  <xsl:template match="employees">  <table border="1" width="400">  <tbody>  <tr>  <th>姓名</th>  <th>年龄</th>  <th>月薪</th>  </tr>  <xsl:for-each select="employee">  <xsl:sort order="ascending" select="name"/>  ***<xsl:if test="age&lt;30">***  ***<tr>***  ***<td>***  ***<xsl:value-of select="name"/>***  ***</td>***  ***<td>***  ***<xsl:value-of select="age"/>***  ***</td>***  ***<td>***  ***<xsl:value-of select="monthly\_pay"/>***  ***</td>***  ***</tr>***  ***</xsl:if>***  </xsl:for-each>  </tbody>  </table>  </xsl:template>  </xsl:stylesheet> |

显示结果如图7所示：

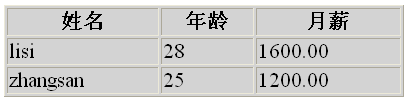


图7 例8显示结果

#### <xsl:choose>元素

<xsl:choose>元素通过<xsl：when>和<xsl:otherwise>一起配合是用来表达多条件语句。

语法：

|  |
| --- |
| <xsl:choose>  <xsl:when test="expression">  ... some output ...  </xsl:when>  <xsl:otherwise>  ... some output ....  </xsl:otherwise>  </xsl:choose> |

示例9：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  <xsl:template match="/">  <html>  <head>  <title>雇员月薪表</title>  <style type="text/css">  {background-color="lightgrey"}  </style>  </head>  <body>  <xsl:apply-templates/>  </body>  </html>  </xsl:template>  <xsl:template match="employees">  <table border="1" width="400">  <tbody>  <tr>  <th>姓名</th>  <th>年龄</th>  <th>月薪</th>  </tr>  <xsl:for-each select="employee">  <xsl:sort order="ascending" select="name"/>  <tr>  ***<xsl:choose>***  ***<xsl:when test="age&lt;25">***  ***<td bgcolor="lightblue">***  ***<xsl:value-of select="name"/>***  ***</td>***  ***</xsl:when>***  ***<xsl:when test="age&gt;25">***  ***<td bgcolor="darkslateblue">***  ***<xsl:value-of select="name"/>***  ***</td>***  ***</xsl:when>***  ***<xsl:otherwise>***  ***<td>***  ***<xsl:value-of select="name"/>***  ***</td>***  ***</xsl:otherwise>***  ***</xsl:choose>***  <td>  <xsl:value-of select="age"/>  </td>  <td>  <xsl:value-of select="monthly\_pay"/>  </td>  </tr>  </xsl:for-each>  </tbody>  </table>  </xsl:template>  </xsl:stylesheet> |

显示结果如图8所示：

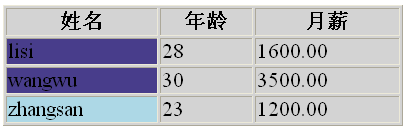


图8 例9显示结果

#### <xsl:copy>元素

该元素用于从原文档树中复制出通过标记匹配模式指定的节点和其文本节点，但是不会复制该节点的属性节点和子孙节点。

|  |
| --- |
| **说明：**使用该元素主要是为了改变现有XML文档的结构或内容，并不适合用于进行显示格式的控制。另外，通常使用该元素的XSL的文档应用到某个XML文档，通过浏览器并不会显示出复制的节点元素，而只显示对应该节点元素的文本节点子元素。 |

#### <xsl:copy­-of>元素

该元素与<xsl:copy>元素的不同是，它除了会从原文档树中复制出通过标记匹配模式指定的节点外，还会将其属性节点和子孙节点都复制到目标文档树中。

* + - **匹配节点的模式**

**（1）匹配根节点**

<xsl:template match＝”/”>

**（2）匹配元素节点**

<xsl:template match＝”employees”>

**（3）使用通配符－－使用（\*）作为通配符**

<xsl:template match＝”\*”>

表示匹配任何元素

**（4）使用@匹配属性：**

XSLT处理器在遍历节点的树状结构是，并不处理属性节点。如果要让处理器处理属性节点，必须显式的调用<xsl:apply-templates>元素，并使用select属性选择属性节点。

<xsl:template match="employee">

<xsl:apply-templates select="@sn" />

</xsl:template>

<xsl:template match="@sn">

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

</xsl:template>

**（5）使用斜杠(/)匹配子节点**

<xsl:template match="employees/employee">

表示匹配employees元素节点下的任意的employee子节点。

<xsl:template match="employees/employee/name">

表示匹配employees元素节点下的任意的employee子节点下的任意name子节点

**（6）使用双斜杠(//)匹配后代节点**

有时候，用户希望对某一类元素节点进行处理，而不管其他节点在树状结构中所处的位置，就可以使用双斜杠(//)来匹配节点。

<xsl:template match=“//name">

匹配根节点下的任意name元素节点，不管该节点所处在的位置

<xsl:template match=“employees//name">

匹配employees元素节点下的任意name元素节点。

**（7）使用（.）匹配当前节点&使用（..）匹配当前节点的双亲节点**

<xsl:template match=" monthly\_pay ">

<tr>

<td><xsl:value-of select=“../name"/></td>

<td><xsl:value-of select=“../age"/></td>

<td><xsl:value-of select=“."/></td>

</tr>

</xsl:template>

**（8）使用或操作符(|)**

使用或操作符(|)来可选的匹配多个节点。

<xsl:template match=“name|age">

匹配任意的name或age元素节点。

<xsl:template match=“employee/name|monthly\_pay/@mode">

匹配employee元素节点下的任意的name子结点或monthly\_pay元素节点的mode属性节点。

**（9）使用[ ]操作符**

用户可以在[ ]中指定布尔表达式，用于在节点集中更精确的匹配某一个节点。

<xsl:template match=“employee[1]”> 等价于

<xsl:template match=“employee[position()=1]”>

匹配第一个employee子元素节点。

**三、实验内容**

1. 请编写一个XSL文档应用于下表中的wpqd.XML文档中，将其转换得到如下图显示的结果。

|  |
| --- |
| <?xml version="1.0"?>  <order>  <order-number>0001</order-number>  <item-info>  <item-name>计算器</item-name>  <quantity>5</quantity>  <price>20 元</price>  </item-info>  <item-info>  <item-name>记事本</item-name>  <quantity>20</quantity>  <price>3 元</price>  </item-info>  <item-info>  <item-name>钢笔</item-name>  <quantity>10</quantity>  <price>15 元</price>  </item-info>  </order> |

要求显示结果如图9所示：

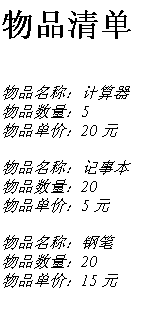


图9 要求实现的显示结果

1. 请编写一个XSL文档应用于下面的student.XML文档中，将其转换得到如图10显示的结果。

|  |
| --- |
| <?xml version="1.0"?>  <roster>  <student ID="101">  <name>李华</name>  <sex>男</sex>  <birthday>1978.9.12</birthday>  <score>92</score>  <skill>Java</skill>  <skill>Oracle</skill>  <skill>C Sharp</skill>  <skill>SQL Server</skill>  </student>  <student ID="102">  <name>倪冰</name>  <sex>女</sex>  <birthday>1979.1.12</birthday>  <score>89</score>  <skill>Visual Basic</skill>  <skill>SQL Server</skill>  <skill>ASP</skill>  </student>  <student ID="103">  <name>张君宝</name>  <sex>男</sex>  <birthday>1982.9.9</birthday>  <score>100</score>  <skill>C Sharp</skill>  <skill>SQL Server</skill>  <skill>UML</skill>  </student>  <student ID="104">  <name>杨惠</name>  <sex>女</sex>  <birthday>1980.5.16</birthday>  <score>85</score>  <skill>Visual C++</skill>  <skill>SQL Server</skill>  <skill>UML</skill>  </student>  <student ID="105">  <name>崔春晓</name>  <sex>男</sex>  <birthday>1981.4.19</birthday>  <score>86</score>  <skill>UML</skill>  <skill>C Sharp</skill>  <skill>XML</skill>  <skill>SQL Server</skill>  </student>  <student ID="106">  <name>王动</name>  <sex>男</sex>  <birthday>1981.4.19</birthday>  <score>59</score>  <skill>UML</skill>  <skill>C Sharp</skill>  <skill>XML</skill>  <skill>SQL Server</skill>  </student>  <student ID="107">  <name>李寻欢</name>  <sex>男</sex>  <birthday>1981.4.19</birthday>  <score>58</score>  <skill>UML</skill>  <skill>C Sharp</skill>  <skill>XML</skill>  <skill>SQL Server</skill>  </student>  </roster> |



图10 要求实现的显示结果

**四、作业**

根据提供的student.xml文件，编写对应的XSL文件，要求：

学生信息以表格形式显示；

按照每位学生的成绩score作降序排序；

学生的ID属性为偶数，所在表格行带有底色；

学生性别为女时，对应的学生姓名以蓝色显示；

表格最后一个列为“评价”，该列以score为标准转换。低于60为不及格，60-80为一般，80-90为良好，90以上为优秀；

表格最后三列统计显示学生的总分、总人数及平均分。

最后统计60分以下的不及格人数及85分以上女生人数。

将其转换得到如图11显示的结果。



图11 作业要求实现的显示结果