**XSL转换(XSLT)**

XSL转换机制可以**指定**将XML文档转换为其他格式的**规则**，例如，txt纯文本、XHTML或其他任何XML格式。

用途：

XSLT通常用来将某种机器可读的XML格式转译为另一种机器可读的XML格式，或者将XML转译为适合人类阅读的表示格式

步骤：

1. 需要提供**XSLT样式表**，它描述了XML文档向某种格式转换的规则
2. XSLT处理器读入**XML文档**和**XSLT样式表**，产生所要的输出

**XSLT样式表：**

<?xml version=”1.0” encoding=”utf-8”?>

<xsl:stylesheet

Xmlns:xsl=<http://www.w3c.org/1999/XSL/Transform>

Version=”1.0”>

<xsl:output method=”html”/> //指定输出的格式为HTML，其他有XML、text

Template1 模板

.

.

.

Template2

</xsl:stylesheet>

**模板示例:**

<xsl:template match=”/staf/employee”>

<tr><xsl:apply-templates/></tr>

</xsl:template>

match属性的值是一个**XPath**表达式

该模板表示：每当看懂啊XPath集/staff/employee中的一个节点时

1. 产生字符串<tr>
2. 对于要处理的**子元素**继续应用模板（会递归！）
3. 当处理完所有所有子元素后，产生字符串</tr>

把属性值复制到输出中的模板：

<xsl:template match=”/staff/employee/hiredate”>

<td><xsl:**value-of** **select**=”@year”/>-<xsl:value-of select=”@month”/>-<xsl:value-of select=”@day”/></td>

</xsl:template>

这里xsl:value-of语句用于计算，由select的XPath值指定的**节点集（属性节点）**的，字符串值

**XML转HTML示例：**

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

<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

version="1.0">

<xsl:output method="html"/> //指定输出为html格式

<xsl:template match="/staff"> //使用XPath来分别定位

<table border="1"><xsl:apply-templates/></table>

</xsl:template>

<xsl:template match="/staff/employee"> //使用XPath来分别定位

<tr><xsl:apply-templates/></tr>

</xsl:template>

<xsl:template match="/staff/employee/name"> //使用XPath来分别定位

<td><xsl:apply-templates/></td>

</xsl:template>

<xsl:template match="/staff/employee/salary"> //使用XPath来分别定位

<td>$<xsl:apply-templates/></td>

</xsl:template>

<xsl:template match="/staff/employee/hiredate"> //使用XPath来分别定位

<td><xsl:value-of select="@year"/>-<xsl:value-of

select="@month"/>-<xsl:value-of select="@day"/></td>

</xsl:template>

</xsl:stylesheet>

**属性文件格式的纯文本:**

<?xml version="1.0"?>

<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

version="1.0">

<xsl:output method="text" omit-xml-declaration="yes"/>

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

employee.<xsl:value-of select="position()" //注意中间生造了文本employee.

/>.name=<xsl:value-of select="name/text()"/>

employee.<xsl:value-of select="position()"

/>.salary=<xsl:value-of select="salary/text()"/>

employee.<xsl:value-of select="position()"

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

/>-<xsl:value-of select="hiredate/@month"

/>-<xsl:value-of select="hiredate/@day"/>

</xsl:template>

</xsl:stylesheet>

Java平台下的XSL转换（XSLT）：

（1）为每个样式表设置一个转换器工厂，得到一个转换器对象，并让它把一个源转换成结果

File styleSheet = new File(fileName); //某个xsl文件

StreamSource styleSource = new StreamSource(styleSheet);

Transformer t = TransformerFactory.newInstance().newTransformer(styleSource);

t.transform(source, result);

**（2）谈谈source和 result**

Source接口有四个实现类：DOMSource、SAXSource、StAXSource、StreamSource，这些类使我们能通过文件、流、Reader、URL、DOM树节点来传入一个source

比如：来自DOM树节点的DOMSource（传入Document就相当于整个树）

Result接口有三个实现类：DOMResult、SAXResult、StreamResult

如果要把转换结果存储到DOM树中，那就再创建一个新的文档节点包装到DOMResult中：

Document doc = builder.newDocument();

t.transform(source, new DOMResult(doc));

如果要把转换结果保存到文件中，可以使用StreamResult

t.transform(source, new StreamResult(file));

**题外话： 将非XML的遗留数据转换成XML的一个小技巧**

Source中的SAXSource可以 从XMLReader接口的SAX实现获得！可以不必从一个现有XML文件开始工作。

先上代码：

t.transform(new SAXSource(new EmployeeReader(), new InputSource(new FileInputStream(filename))), result);

其中EmployeeReader就是XMLReader的SAX实现类，InputSource就是文件来源，result决定保存的形式

当然，大多数XSLT应用程序都已经有了XML格式的输入数据，此时只需在一个StreamSource对象上调用transform方法即可：

t.transform(new StreamSource(file),  result);