XSLT 教程

XSL 指扩展样式表语言（EXtensible Stylesheet Language），它是一个 XML 文档的样式表语言。

XSLT 指 XSL 转换。在此教程中，你将学习如何使用 XSLT 将 XML 文档转换为其他文档，比如 XHTML。

现在开始学习 **XSLT！**

XSLT 实例

XSLT 实例

```
<?xml version="1.0"?>
<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>
```

尝试一下 »

XSLT 参考手册

XSLT 元素

对所有来自 W3C 标准的 XSLT 元素进行了描述，以及关于浏览器支持的信息。

XSLT 函数

XSLT 包含了超过 100 个内置函数。 这些函数可以用于字符串、数值、日期和时间比较、节点和 QName 处理, 序列处理, 逻辑值等等。

XSL 语言

它起始于 XSL，结束于 XSLT、XPath 以及 XSL-FO。

起始于 XSL

XSL 指扩展样式表语言（EXtensible Stylesheet Language）。

万维网联盟（W3C）开始发展 XSL 的原因是：存在着对于基于 XML 的样式表语言的需求。

CSS = HTML 样式表

HTML 使用预先定义的标签，每个标签的意义**很容易被理解**。

HTML 中的 <table> 标签定义表格 - 并且浏览器**清楚如何显示它**。

向 HTML 元素添加样式是很容易的。通过 CSS，很容易告知浏览器用特定的字体或颜色显示一个元素。

XSL = XML 样式表

XML 不使用预先定义的标签（我们可以使用任何喜欢的标签名），并且每个标签的意义**并不都那么容易被理解**。

<table> 标签意味着一个 HTML 表格，一件家具，或是别的什么东西 - 浏览器**不清楚如何显示它**。

XSL 可描述如何来显示 XML 文档！

XSL - 不仅仅是样式表语言

XSL 包括三部分：

XSLT - 一种用于转换 XML 文档的语言。

XPath - 一种用于在 XML 文档中导航的语言。

XSL-FO - 一种用于格式化 XML 文档的语言。

本教程的主要内容是 XSLT

本教程的其余部分是 XSLT - 用来转换 XML 文档的语言。

如需学习更多有关 XPath 和 XSL-FO 的知识，请访问我们的XPath 教程 和 XSL-FO 教程。

XSLT 简介

XSLT 是一种用于将 XML 文档转换为 XHTML 文档或其他 XML 文档的语言。

XPath 是一种用于在 XML 文档中进行导航的语言。

您需要具备的基础知识

在您继续学习之前，需要对以下知识有基本的了解：

- HTML / XHTML
- XML / XML 命名空间
- XPath

如果您想要首先学习这些项目，请在我们的首页访问这些教程。

什么是 XSLT?

- XSLT 指 XSL 转换（XSL Transformations）
- XSLT 是 XSL 中最重要的部分
- XSLT 可将一种 XML 文档转换为另外一种 XML 文档
- XSLT 使用 XPath 在 XML 文档中进行导航
- XSLT 是一个 W3C 标准

XSLT = XSL 转换

XSLT 是 XSL 中最重要的部分。

XSLT 用于将一种 XML 文档转换为另外一种 XML 文档，或者可被浏览器识别的其他类型的文档，比如 HTML 和 XHTML。通常，XSLT 是通过把每个 XML 元素转换为 (X)HTML 元素来完成这项工作的。

通过 XSLT，您可以向输出文件添加元素和属性，或从输出文件移除元素和属性。您也可重新排列并分类元素，执行测试并决定隐藏或显示哪个元素，等等。

描述转化过程的一种通常的说法是，**XSLT 把 XML 源树转换为 XML 结果树**。

XSLT 使用 XPath

XSLT 使用 XPath 在 XML 文档中查找信息。XPath 被用来通过元素和属性在 XML 文档中进行导航。

如果您想要首先学习 XPath，请访问我们的 XPath 教程。

它如何工作？

在转换过程中，XSLT 使用 XPath 来定义源文档中可匹配一个或多个预定义模板的部分。一旦匹配被找到，XSLT 就会把源文档的匹配部分转换为结果文档。

XSLT 是一个 W3C 标准

XSLT 在 1999 年 11 月 16 日被确立为 W3C 标准。

如需更多有关 W3C 的 XSLT 活动的信息，请访问我们的 W3C 教程。

XSLT 浏览器

所有主流的浏览器都支持 XML 和 XSLT。

Mozilla Firefox

从版本 3 开始，Firefox 就已支持 XML、XSLT 和 XPath。

Internet Explorer

从版本 6 开始，Internet Explorer 就已支持 XML、XSLT 和 XPath。

Internet Explorer 5 不兼容官方的 W3C XSL 标准。

Google Chrome

从版本 1 开始，Chrome 就已支持 XML、XSLT 和 XPath。

Opera

从版本 9 开始，Opera 就已支持 XML、XSLT 和 XPath。Opera 8 仅支持 XML + CSS。

Apple Safari

从版本 3 开始，Safari 就已支持 XML 和 XSLT。

反馈/建议

Copyright © 2013-2018 菜鸟教程 runoob.com All Rights Reserved. 备案号：闽ICP备15012807号-1

首页 HTML CSS JS 本地书签

XSLT 浏览器 XSLT <xsl:template> 元素

XSLT - 转换

实例研究：如何使用 XSLT 将 XML 转换为 XHTML。

我们会在下一章对本实例的细节进行解释。

正确的样式表声明

把文档声明为 XSL 样式表的根元素是 <xsl:stylesheet> 或 <xsl:transform>。

注意： <xsl:stylesheet> 和 <xsl:transform> 是完全同义的，均可被使用！

根据 W3C 的 XSLT 标准，声明 XSL 样式表的正确方法是：

```
<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">
```

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。

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"。

从一个原始的 XML 文档开始

我们现在要把下面这个 XML 文档（"cdcatalog.xml"）转换为 XHTML：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
<cd>
<title>Empire Burlesque</title>
<artist>Bob Dylan</artist>
<country>USA</country>
<company>Columbia</company>
<price>10.90</price>
<year>1985</year>
</cd>
.
.
</catalog>
```

在 Firefox 和 Internet Explorer 中查看 XML 文件： 打开 XML 文件（通常通过点击某个链接）- XML 文档会以颜色化的代码方式来显示根元素及子元素。点击元素左侧的加号（+）或减号（-）可展开或收缩元素的结构。如需查看原始的 XML 源文件（不带有加号和减号），请在浏览器菜单中选择"查看页面源代码"或"查看源代码"。

在 Netscape 6 中查看 XML 文件： 打开 XML 文件，然后在 XML 文件中右击，并选择"查看页面源代码"。XML 文档会以颜色化的代码方式来显示根元素及子元素。

在 Opera 7 中查看 XML 文件： 打开 XML 文件，然后在 XML 文件中右击，选择"框架"/"查看源代码"。XML 文档将显示为纯文本。

查看 "cdcatalog.xml"

创建 XSL 样式表

然后创建一个带有转换模板的 XSL 样式表（"cdcatalog.xsl"）：

```
<?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>
```

查看 "cdcatalog.xsl"

把 XSL 样式表链接到 XML 文档

向 XML 文档（"cdcatalog.xml"）添加 XSL 样式表引用：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.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>
.
.
</catalog>
```

如果您使用的浏览器兼容 XSLT，它会很顺利地把您的 XML 转换为 XHTML。

查看结果

我们会在下一章对上面的例子中的细节进行解释。

XSLT 浏览器　　XSLT <xsl:template> 元素

XSLT 转换　　XSLT <xsl:value-of> 元素

XSLT <xsl:template> 元素

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。

<xsl:template> 元素

<xsl:template> 元素用于构建模板。

match 属性用于关联 XML 元素和模板。match 属性也可用来为整个 XML 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）。

好了，让我们看一下上一章中的 XSL 文件的简化版本：

实例

```
<?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>
<body>
<h2>My CD Collection</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th>Title</th>
<th>Artist</th>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

尝试一下 »

实例解释

由于 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"?>**.

下一个元素，**<xsl:stylesheet>**，，定义此文档是一个 XSLT 样式表文档（连同版本号和 XSLT 命名空间属性）。

<xsl:template> 元素定义了一个模板。而 **match="/"** 属性则把此模板与 XML 源文档的根相联系。

<xsl:template> 元素内部的内容定义了写到输出结果的 HTML 代码。

最后两行定义了模板的结尾及样式表的结尾。

这个实例的结果有一点小缺陷，因为数据没有从 XML 文档被复制到输出。在下一章中，您将学习到如何使用 **<xsl:value-of>** 元素从 XML 元素选取值。

XSLT 转换　　XSLT <xsl:value-of> 元素

点我分享笔记

反馈/建议

XSLT <xsl:template> 元素　　XSLT <xsl:for-each> 元素

XSLT <xsl:value-of> 元素

<xsl:value-of> 元素用于提取某个选定节点的值。

<xsl:value-of> 元素

<xsl:value-of> 元素用于提取某个 XML 元素的值，并把值添加到转换的输出流中：

实例

```
<?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>
<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>
```

尝试一下 »

实例解释

注意：在上面的实例中，**select** 属性的值是一个 XPath 表达式。这个 XPath 表达式的工作方式类似于定位某个文件系统，在其中正斜杠（/）可选择子目录。

上面实例的结果有一点小缺陷，仅有一行数据从 XML 文档被复制到输出。在下一章中，您将学习到如何使用 **<xsl:for-each>** 元素来循环遍历 XML 元素，并显示所有的记录。

XSLT <xsl:template> 元素　　XSLT <xsl:for-each> 元素

点我分享笔记

反馈/建议

XSLT <xsl:value-of> 元素　　XSLT <xsl:sort> 元素

XSLT <xsl:for-each> 元素

<xsl:for-each> 元素允许您在 XSLT 中进行循环。

<xsl:for-each> 元素

XSL <xsl:for-each> 元素可用于选取指定的节点集中的每个 XML 元素：

实例

```
<?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>
<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>
```

尝试一下 »

注意：select 属性的值是一个 XPath 表达式。这个 XPath 表达式的工作方式类似于定位某个文件系统，在其中正斜杠（/）可选择子目录。

过滤输出结果

通过在 <xsl:for-each> 元素中添加一个选择属性的判别式，我们也可以过滤从 XML 文件输出的结果。

<xsl:for-each select="catalog/cd[artist='Bob Dylan']">

合法的过滤运算符:

= （等于）

!= （不等于）

< （小于）

> （大于）

看看调整后的 XSL 样式表：

实例

```
<?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>
<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[artist='Bob Dylan']">
<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>
```

尝试一下 »

XSLT <xsl:sort> 元素

<xsl:sort> 元素用于对输出结果进行排序。

在何处放置排序信息

如需对输出结果进行排序，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：

实例

```
<?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>
<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>
```

尝试一下 »

注意：**select** 属性指示需要排序的 XML 元素。

XSLT <xsl:if> 元素

<xsl:if> 元素用于放置针对 XML 文件内容的条件测试。

<xsl:if> 元素

如需放置针对 XML 文件内容的条件测试，请向 XSL 文档添加 <xsl:if> 元素。

语法

```
<xsl:if test="expression">
...如果条件成立则输出...
</xsl:if>
```

在何处放置 <xsl:if> 元素

如需添加有条件的测试，请在 XSL 文件中的 <xsl:for-each> 元素内部添加 <xsl:if> 元素：

实例

```
<?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>
<body>
<h2>My CD Collection</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th>Title</th>
<th>Artist</th>
<th>Price</th>
</tr>
<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>
<td><xsl:value-of select="price"/></td>
</tr>
</xsl:if>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

尝试一下 »

注意：必需的 **test** 属性的值包含了需要求值的表达式。

上面的代码仅仅会输出价格高于 10 的 CD 的 title 和 artist 元素。

XSLT <xsl:sort> 元素 XSLT <xsl:choose> 元素

XSLT <xsl:if> 元素 XSLT <xsl:apply-templates> 元素

XSLT <xsl:choose> 元素

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

<xsl:choose> 元素

语法

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

在何处放置选择条件

如需插入针对 XML 文件的多重条件测试，请向 XSL 文件添加 <xsl:choose>、<xsl:when> 以及 <xsl:otherwise> 元素：

实例

```
<?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>
<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>
```

尝试一下 »

上面的代码会在 CD 的价格高于 10 时向 "Artist" 列添加粉色的背景颜色。

另一个实例

这是另外一个包含两个 <xsl:when> 元素的实例：

实例

```
<?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>
<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:when test="price &gt; 9">
<td bgcolor="#cccccc">
<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>
```

尝试一下 »

上面的代码会在 CD 的价格高于 10 时向 "Artist" 列添加粉色的背景颜色，并在 CD 的价格高于 9 且低于等于 10 时向 "Artist" 列添加灰色的背景颜色。

XSLT <xsl:apply-templates> 元素

<xsl:apply-templates> 元素可把一个模板应用于当前的元素或者当前元素的子节点。

<xsl:apply-templates> 元素

<xsl:apply-templates> 元素可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。

请看下面的 XSL 样式表：

实例

```
<?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>
<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>
```

尝试一下 »

XSLT - 在客户端

如果您的浏览器支持 XSLT，那么在浏览器中它可被用来将文档转换为 XHTML。

JavaScript 解决方案

在前面的章节，我们已向您讲解如何使用 XSLT 将某个 XML 文档转换为 XHTML。我们是通过以下途径完成这个工作的：向 XML 文件添加 XSL 样式表，并通过浏览器完成转换。

即使这种方法的效果很好，在 XML 文件中包含样式表引用也不总是令人满意的（例如，在无法识别 XSLT 的浏览器这种方法就无法奏效）。

更通用的方法是使用 JavaScript 来完成转换。

通过使用 JavaScript，我们可以：

进行浏览器确认测试

根据浏览器和用户需求来使用不同的样式表

这就是 XSLT 的魅力所在！XSLT 的设计目的之一就是使数据从一种格式转换到另一种格式成为可能，同时支持不同类型的浏览器以及不同的用户需求。

客户端的 XSLT 转换一定会成为未来浏览器所执行的主要任务之一，同时我们也会看到其在特定的浏览器市场的增长（盲文、听觉浏览器、网络打印机，手持设备，等等）。

XML 文件和 XSL 文件

请看这个在前面的章节已展示过的 XML 文档：

```
<?xml version="1.0" encoding="UTF-8"?>

<catalog>

<cd>

<title>Empire Burlesque</title>

<artist>Bob Dylan</artist>

<country>USA</country>

<company>Columbia</company>

<price>10.90</price>
```

```
<year>1985</year>

</cd>

.

.

</catalog>
```

查看 XML 文件。

以及附随的 XSL 样式表：

```
<?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>

<body>

  <h2>My CD Collection</h2>

  <table border="1">

    <tr bgcolor="#9acd32">

      <th style="text-align:left">Title</th>

      <th style="text-align:left">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>
```

查看 XSL 文件。

请注意，这个 XML 文件没有包含对 XSL 文件的引用。

重要事项：上面这句话意味着，XML 文件可使用多个不同的 XSL 样式表来进行转换。

在浏览器中把 XML 转换为 XHTML

这是用于在客户端把 XML 文件转换为 XHTML 的源代码：

实例

```
<!DOCTYPE html>
<html>
<head>
<script>
function loadXMLDoc(filename)
{
if (window.ActiveXObject)
{
xhttp = new ActiveXObject("Msxml2.XMLHTTP");
}
else
{
xhttp = new XMLHttpRequest();
}
xhttp.open("GET", filename, false);
try {xhttp.responseType = "msxml-document"} catch(err) {} // Helping IE11
xhttp.send("");
return xhttp.responseXML;
}
function displayResult()
{
xml = loadXMLDoc("cdcatalog.xml");
xsl = loadXMLDoc("cdcatalog.xsl");
// code for IE
if (window.ActiveXObject || xhttp.responseType == "msxml-document")
{
ex = xml.transformNode(xsl);
document.getElementById("example").innerHTML = ex;
}
// code for Chrome, Firefox, Opera, etc.
else if (document.implementation && document.implementation.createDocument)
{
xsltProcessor = new XSLTProcessor();
xsltProcessor.importStylesheet(xsl);
resultDocument = xsltProcessor.transformToFragment(xml, document);
document.getElementById("example").appendChild(resultDocument);
}
}
</script>
</head>
<body onload="displayResult()">
<div id="example" />
</body>
</html>
```

尝试一下 »

提示：假如您不了解如何编写 JavaScript，请学习我们的 JavaScript 教程。

实例解释：

loadXMLDoc() 函数

loadXMLDoc() 函数是用来加载 XML 和 XSL 文件。

它检查用户拥有的和加载文件的浏览器类型。

displayResult() 函数

该函数用来显示使用 XSL 文件定义样式的 XML 文件。

加载 XML 和 XSL 文件

测试用户拥有的浏览器类型

如果用户浏览器支持 ActiveX 对象：

使用 transformNode() 方法把 XSL 样式表应用到 XML 文档

设置当前文档（id="example"）的 body 包含已经应用样式的 XML 文档

如果用户的浏览器不支持 ActiveX 对象：

创建一个新的 XSLTProcessor 对象并导入 XSL 文件

使用 transformToFragment() 方法把 XSL 样式表应用到 XML 文档

设置当前文档（id="example"）的 body 包含已经应用样式的 XML 文档

XSLT <xsl:apply-templates> 元素 XSLT 在服务器端

点我分享笔记

反馈/建议

XSLT 在客户端 XSLT – 编辑 XML

XSLT - 在服务器端

由于并非所有的浏览器都支持 XSLT，另一种解决方案是在服务器上完成 XML 至 XHTML 的转化。

跨浏览器解决方案

在前面的章节，我们讲解过如何在浏览器中使用 XSLT 来完成 XML 到 XHTML 的转化。我们创建了一段使用 XML 解析器来进行转换的 JavaScript。JavaScript 解决方案无法工作于没有 XML 解析器的浏览器。

为了让 XML 数据适用于任何类型的浏览器，我们必须在服务器上对 XML 文档进行转换，然后将其作为 XHMTL 发送回浏览器。

这是 XSLT 的另一个优点。XSLT 的设计目标之一是使数据在服务器上从一种格式转换到另一种格式成为可能，并向所有类型的浏览器返回可读的数据。

XML 文件和 XSLT 文件

请看这个在前面的章节已展示过的 XML 文档：

```
<?xml version="1.0" encoding="UTF-8"?>

<catalog>

  <cd>

    <title>Empire Burlesque</title>

    <artist>Bob Dylan</artist>

    <country>USA</country>
```

```
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
</catalog>
```

查看 XML 文件。

以及附随的 XSL 样式表：

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

<xsl:template match="/">
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th style="text-align:left">Title</th>
      <th style="text-align:left">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>
</xsl:template>

</xsl:stylesheet>
```

查看 XSL 文件。

请注意，这个 XML 文件没有包含对 XSL 文件的引用。

重要事项：上面这句话意味着，XML 文件可使用多个不同的 XSL 样式表来进行转换。

在服务器把 XML 转换为 XHTML

这是用于在服务器上把 XML 文件转换为 XHTML 的源代码：

使用 PHP 代码转换：

```
<?php

// 载入 XML 文件

$xml = new DOMDocument;

$xml->load('cdcatalog.xml');



// 载入 XSL 文件

$xsl = new DOMDocument;

$xsl->load('cdcatalog.xsl');



// 设置转换

$proc = new XSLTProcessor;



// 添加 xsl 规则

$proc->importStyleSheet($xsl);



echo $proc->transformToXML($xml);

?>
```

提示：假如您不了解如何编写 php，您可以学习我们的 PHP 教程。

使用 ASP 代码转换：

```
<%

'载入 XML 文件

set xml = Server.CreateObject("Microsoft.XMLDOM")

xml.async = false

xml.load(Server.MapPath("cdcatalog.xml"))



'载入 XSL 文件
```

```
set xsl = Server.CreateObject("Microsoft.XMLDOM")

xsl.async = false

xsl.load(Server.MapPath("cdcatalog.xsl"))



'转换文件

Response.Write(xml.transformNode(xsl))

%>
```

提示：假如您不了解如何编写 ASP，您可以学习我们的 ASP 教程。

第一段代码创建了微软的 XML 解析器（XMLDOM）的一个实例，并把 XML 文件载入了内存。第二段代码创建了解析器的另一个实例，并把这个 XSL 文件载入了内存。最后一行代码使用 XSL 文档转换了 XML 文档，并把结果作为 XHTML 发送到您的浏览器。太好了！

它是如何工作的。

XSLT 在客户端 XSLT – 编辑 XML

XSLT 在服务器端 XML 编辑器

XSLT - 编辑 XML

存储在 XML 文件中的数据可通过因特网浏览器进行编辑。

打开、编辑并保存 XML

现在，我们会为您展示如何打开、编辑及保存存储于服务器上的 XML 文件。

我们将使用 XSL 把 XML 文档转换到一个 HTML 表单中。XML 元素的值会被写到 HTML 表单中的 HTML 输入域。这个 HTML 表单是可编辑的。在被编辑完成后，数据会被提交回服务器，XML 文件会得到更新（这部分由 ASP 完成）。

XML 文件和 XSL 文件

首先，请看将被使用的 XML 文档（"tool.xml"）：

```
<?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>
```

查看 XML 文件。

接着，请看下面的样式表（"tool.xsl"）：

```
<?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.html">
<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>
```

查看 XSL 文件。

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。XML "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。每个 XML "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。

然后，我们还有第二个样式表："tool_updated.xsl"。这个 XSL 文件会被用来显示已更新的 XML 数据。这个样式表不会输出可编辑 HTML 表单，而是一个静态的 HTML 表格：

```
<?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>Updated Tool Information:</h2>
<table border="1">
<xsl:for-each select="tool/field">
<tr>
<td><xsl:value-of select="@id" /></td>
<td><xsl:value-of select="value" /></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>

</xsl:stylesheet>
```

查看 XSL 文件。

ASP 文件

在上面 "tool.xsl" 文件中，HTML 表单的 action 属性的值是 "edittool.asp" 。

"edittool.asp" 页面包含两个函数：loadFile() 函数载入并转换 XML 文件，updateFile() 函数更新 XML 文件：

```
<%
function loadFile(xmlfile,xslfile)
Dim xmlDoc,xslDoc
'Load XML file
set xmlDoc = Server.CreateObject("Microsoft.XMLDOM")
xmlDoc.async = false
xmlDoc.load(xmlfile)
'Load XSL file
set xslDoc = Server.CreateObject("Microsoft.XMLDOM")
xslDoc.async = false
xslDoc.load(xslfile)
'Transform file
Response.Write(xmlDoc.transformNode(xslDoc))
end function

function updateFile(xmlfile)
Dim xmlDoc,rootEl,f
Dim i
'Load XML file
set xmlDoc = Server.CreateObject("Microsoft.XMLDOM")
xmlDoc.async = false
xmlDoc.load(xmlfile)

'Set the rootEl variable equal to the root element
Set rootEl = xmlDoc.documentElement

'Loop through the form collection
for i = 1 To Request.Form.Count
'Eliminate button elements in the form
if instr(1,Request.Form.Key(i),"btn_")=0 then
'The selectSingleNode method queries the XML file for a single node
'that matches a query. This query requests the value element that is
'the child of a field element that has an id attribute which matches
'the current key value in the Form Collection. When there is a match -
'set the text property equal to the value of the current field in the
'Form Collection.
set f = rootEl.selectSingleNode("field[@id='" & _
Request.Form.Key(i) & "']/value")
f.Text = Request.Form(i)
end if
next

'Save the modified XML file
xmlDoc.save xmlfile

'Release all object references
set xmlDoc=nothing
set rootEl=nothing
set f=nothing

'Load the modified XML file with a style sheet that
'allows the client to see the edited information
loadFile xmlfile,server.MapPath("tool_updated.xsl")
end function

'If the form has been submitted update the
'XML file and display result - if not,
'transform the XML file for editing
if Request.Form("btn_sub")="" then
loadFile server.MapPath("tool.xml"),server.MapPath("tool.xsl")
else
updateFile server.MapPath("tool.xml")
end if
%>
```

提示：假如您不了解如何编写 ASP，请学习我们的 ASP 教程。

注意：我们正在转换并更新位于服务器上的 XML 文件。这是一个跨平台的解决方案。客户端仅能获得从服务器返回的 HTML - 而 HTML 可运行于任何浏览器。

XML 编辑器

如果希望极认真地学习和使用 XML，那么您一定会从使用一款专业的 XML 编辑器中受益。

XML 是基于文本的

XML 是基于文本的标记语言。

关于 XML 的一件很重要的事情是，XML 文件可被类似记事本这样的简单的文本编辑器来创建和编辑。

不过，在您开始使用 XML 进行工作时，您很快会发现，使用一款专业的 XML 编辑器来编辑 XML 文档会更好。

为什么不使用记事本？

许多 Web 开发人员使用记事本来编辑 HTML 和 XML 文档，这是因为最常用的操作系统都带有记事本，而且它很容易使用。从个人来讲，我经常使用记事本来快速地编辑某些简单的 HTML、CSS 以及 XML 文件。

但是，如果您使用记事本对 XML 进行编辑，可能很快会发现不少问题。

记事本不能确定您编辑的文档类型，所以也就无法辅助您的工作。

为什么使用 XML 编辑器？

当今，XML 是非常重要的技术，并且开发项目正在使用这些基于 XML 的技术：

- 用 XML Schema 定义 XML 的结构和数据类型
- 用 XSLT 来转换 XML 数据
- 用 SOAP 来交换应用程序之间的 XML 数据
- 用 WSDL 来描述网络服务
- 用 RDF 来描述网络资源
- 用 XPath 和 XQuery 来访问 XML 数据
- 用 SMIL 来定义图形

为了能够编写出无错的 XML 文档，您需要一款智能的 XML 编辑器！

XML 编辑器

专业的 XML 编辑器会帮助您编写无错的 XML 文档，根据某种 DTD 或者 schema 来验证 XML，以及强制您创建合法的 XML 结构。

XML 编辑器应该具有如下能力：

- 为开始标签自动添加结束标签
- 强制您编写合法的 XML
- 根据 DTD 来验证 XML

根据 Schema 来验证 XML

对您的 XML 语法进行代码的颜色化显示

XSLT – 编辑 XML　　XSLT 总结

XML 编辑器　　XSLT 实例

您已经学习了 XSLT，现在该怎么办？

XSLT 总结

本教程已经向您讲解了如何使用 XSLT 把 XML 文档转换为其他格式，比如 XHTML。

您已经学会如何向输出文件添加元素和属性，并从输出文件移除元素和属性。

您也学习了如何对元素进行重新排列和排序，执行测试以及决定显示或隐藏哪些元素。

如需学习更多有关 XSLT 的知识，请查阅我们的 XSLT 参考手册。

您已经学习了 XSLT，下一步应当学习什么呢？

XSL 包括三种语言：XSLT、XPath 以及 XSL-FO，因此下一步应当学习 XPath 和 XSL-FO。

XPath

XPath 用于经由元素和属性在 XML 文档中进行导航。

XPath 是 W3C 的 XSL 标准的主要元素。对 XPath 的理解是深入使用 XML 的基础。

如果缺乏 XPath 方面的知识，您就无法创建 XSLT 文档。

如果你希望学习更多有关 XPath 的知识，请访问我们的 XPath 教程。

XSL-FO

XSL-FO 可描述输出到屏幕、纸媒介或者其他媒介的 XML 数据的格式。

XSL-FO 文档是带有关于输出布局以及输出内容的信息的 XML 文件。

如果你希望学习更多有关 XSL-FO 的知识，请访问我们的 XSL-FO 教程。

XML 编辑器　　XSLT 实例

XSLT 实例

<xsl:template> 元素

包含当指定节点匹配时要应用的规则的模板

实例解释

<xsl:value-of> 元素

提取选定节点的值

实例解释

<xsl:for-each> 元素

通过 <xsl:for-each> 元素选取指定节点集中的每个 XML 元素

过滤节点集的输出结果

实例解释

<xsl:sort> 元素

排序节点集的输出结果

实例解释

<xsl:if> 元素

放置针对 XML 文件内容的条件测试

实例解释

<xsl:choose> 元素

实例 1

该实例将会在 CD 的价格高于 10 时向 "Artist" 列添加粉色的背景颜色

实例 2

该实例将会在 CD 的价格高于 10 时向 "Artist" 列添加粉色的背景颜色，并在 CD 的价格高于 9 且低于等于 10 时向 "Artist" 列添加灰色的背景颜色

实例解释

<xsl:apply-templates> 元素

把模板应用于元素

实例解释

XSLT 元素参考手册

源自于 W3C 推荐标准（XSLT Version 1.0）的 XSLT 元素。

XSLT 元素

如果您需要有关下列元素的更详细的信息，请点击元素列中的链接。

元素	描述
apply-imports	应用来自导入样式表中的模版规则。
apply-templates	向当前元素或当前元素的子节点应用模板规则。
attribute	添加属性。
attribute-set	定义命名的属性集。
call-template	调用一个指定的模板。
choose	与 <when> 以及 <otherwise> 协同使用，来表达多重条件测试。
comment	在结果树中创建注释节点。
copy	创建当前节点的一个副本（无子节点及属性）。
copy-of	创建当前节点的一个副本（带有子节点及属性）。
decimal-format	定义当通过 format-number() 函数把数字转换为字符串时，所要使用的字符和符号。
element	在输出文档中创建一个元素节点。
fallback	假如处理器不支持某个 XSLT 元素，规定一段替代代码来运行。
for-each	循环遍历指定的节点集中的每个节点。
if	包含一个模板，仅当某个指定的条件成立时应用此模板。
import	用于把一个样式表中的内容导入另一个样式表中。 **注意：** 被导入的样式表的优先级低于导出的样式表。
include	把一个样式表中的内容包含到另一个样式表中。**注意：** 被包含的样式表（included style sheet）拥有与包含的样式表（including style sheet）相同的优先级。
key	声明一个命名的键，该键通过 key() 函数在样式表中使用。
message	向输出写一条消息（用于报告错误）。
namespace-alias	把样式表中的命名空间替换为输出中不同的命名空间。

number	测定当前节点的整数位置，并对数字进行格式化。
otherwise	规定 <choose> 元素的默认动作。
output	定义输出文档的格式。
param	声明一个局部或全局参数。
preserve-space	定义保留空白的元素。
processing-instruction	向输出写一条处理指令，即生成处理指令节点。
sort	对输出进行排序。
strip-space	定义应当删除空白字符的元素。
stylesheet	定义样式表的根元素。
template	当指定的节点被匹配时所应用的规则。
text	向输出写文本，即通过样式表生成文本节点。
transform	定义样式表的根元素。
value-of	提取选定节点的值。
variable	声明局部或者全局的变量。
when	规定 <choose> 元素的动作。
with-param	规定传递给模板的参数的值。

XSLT <xsl:with-param> 元素 XSLT current() 函数

点我分享笔记

反馈/建议

XSLT unparsed-entity-uri() 函数

XSLT 函数

XQuery 1.0、XPath 2.0 以及 XSLT 2.0 共享相同的函数库。

XSLT 函数

XSLT 含有超过 100 个内建的函数。这些函数用于字符串值、数值、日期和时间比较、节点和 QName 操作、序列操作、布尔值，等等。

函数命名空间的默认前缀是 fn。

函数命名空间的 URI 是： http://www.w3.org/2005/xpath-functions

提示：函数在被调用时常带有 fn: 前缀，比如 fn:string()。 不过，既然 fn: 是命名空间的默认前缀，那么在被调用时，函数的名称不必使用前缀。

您可以在我们的 XPath 教程中访问所有内建的 XSLT 2.0 函数的参考手册。

此外，下面列出了内建的 XSLT 函数：

名称	描述
current()	返回当前节点。
document()	用于访问外部 XML 文档中的节点。
element-available()	检测 XSLT 处理器是否支持指定的元素。
format-number()	把数字转换为字符串。
function-available()	检测 XSLT 处理器是否支持指定的函数。
generate-id()	返回唯一标识指定节点的字符串值。
key()	通过使用由 <xsl:key> 元素规定的索引号返回节点集。
system-property()	返回系统属性的值。
unparsed-entity-uri()	返回未解析实体的 URI。