

1 Linguaxe de transformacións de documentos XML

1.1 Sumario

- 1 Introducción a XSLT
- 2 Linguaxes XSL
- 3 Navegadores XSLT
- 4 Transformacións XSLT
 - ◆ 4.1 Declaración da folia de estilo XSL
 - ◆ 4.2 O documento XML de exemplo
 - ◆ 4.3 Creación dunha folia de estilos XSL
 - ◆ 4.4 Enlace á folia de estilo XSL dende o arquivo XML
- 5 Elemento <xsl:template>
- 6 Elemento <xsl:value-of>
- 7 Elemento <xsl:for-each>
 - ◆ 7.1 Filtrando a saída
- 8 Elemento <xsl:sort>
- 9 Elemento <xsl:if>
- 10 Elemento <xsl:choose>
- 11 Elemento <xsl:apply-templates>

1.2 Introducción a XSLT

XSL son as siglas de *EXtensible Stylesheet Language*, e é unha linguaxe de follas de estilos para os documentos XML.

XSLT quere dicir *XSL Transformations* e é a linguaxe que empregaremos para transformar documentos XML noutros formatos como XHTML.

Co XSLT poderemos engadir/eliminar elementos e atributos dun arquivo orixe XML ou ben no arquivo que obteñamos na saída. Tamén poderíamos reorganizar os elementos, avaliar campos, ocultar elementos, etc.

XSLT é unha recomendación do W3C dende o 16 de novembro de 1999.

1.3 Linguaxes XSL

A transformación de arquivos XML noutros formatos comezou co emprego de XSL e terminou con XSLT, XPath e XSL-FO.

Poderíamos dicir que XSL é ao XML o que as follas de estilo CSS son ao HTML.

XSL = Style Sheets para XML

Por tanto, XSL describe como vai ser amosado o documento XML.

O XSL é máis que unha linguaxe de folia de estilos. O XSL consiste en tres partes:

- XSLT - a linguaxe para transformar documentos XML
- XPath - a linguaxe para navegar nos documentos XML
- XSL-FO - a linguaxe para formatar documentos XML

1.4 Navegadores XSLT

A maior parte dos navegadores teñen soporte para XML e XSLT.

- **Mozilla Firefox:** Firefox soporta XML, XSLT, e XPath dende a versión 3.
- **Internet Explorer:** Internet Explorer soporta XML, XSLT, e XPath dende a versión 6. Internet Explorer 5 non é compatible coa recomendación oficial W3C.
- **Google Chrome:** Chrome soporta XML, XSLT, e XPath dende a versión 1.
- **Opera:** Opera soporta XML, XSLT, e XPath dende a versión 9. Opera 8 soporta soamente XML + CSS.
- **Apple Safari:** Safari soporta XML e XSLT dende a versión 3.

1.5 Transformacións XSLT

Neste apartado imos transformar un arquivo XML nun XHTML empregando XSLT. Os detalles do exemplo serán explicados máis adiante.

1.5.1 Declaración da folia de estilo XSL

O elemento raíz que declara que o documento vai ser unha folia de estilos XSL é **<xsl:stylesheet>** ou **<xsl:transform>**. Poderemos empregar calquera das dúas declaracións .

A forma de declarar o arquivo segundo a recomendación da W3C é:

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

ou

```
<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Para ter acceso aos elementos XSLT, atributos e características deberemos declarar o namespace ao principio do documento.

A declaración **xmlns:xsl="http://www.w3.org/1999/XSL/Transform"** apunta directamente ao XSLT namespace oficial do W3C. Si empregamos este namespace deberemos incluír o atributo **version="1.0"**.

1.5.2 O documento XML de exemplo

Queremos transformar o seguinte documento XML ("cdcatalog.xml") nun arquivo 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>
```

[Descargar arquivo orixinal "cdcatalog.xml"](#)

1.5.3 Creación dunha folia de estilos XSL

Agora imos crear unha folia XSL ("cdcatalog.xsl") cun modelo de transformación:

```
<?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>

```

Descargar o arquivo **cdcatalog.xml** completo

1.5.4 Enlace á folia de estilo XSL dende o arquivo XML

Para referenciar a folia de estilos XSL ao documento XML faremos:

```

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

```

Descarga os dous arquivos o .xml e o .xsl

1.6 Elemento <xsl:template>

O elemento **<xsl:template>** é usado para construír modelos.

O atributo **match** úsase para asociar o modelo co elemento XML. O atributo match pode ser usado para definir un modelo para o documento XML.

O valor do atributo match é unha expresión XPath (por exemplo match="/" define o documento completo).

Unha versión simplificada do arquivo XSL anterior pode ser:

```

<?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>.</td>
        <td>.</td>
      </tr>
    </table>
    </body>
    </html>
  </xsl:template>

</xsl:stylesheet>

```

Explicación do Exemplo:

Xa que unha folia XSL é tamén un documento XML, terá que comezar sempre coa declaración XML: **<?xml version="1.0" encoding="ISO-8859-1"?>**

O seguinte elemento, **<xsl:stylesheet>** define que este documento é unha folia XSLT (co seu número de versión e atributo namespace).

O elemento **<xsl:template>** define un modelo. O atributo **match="/"** asocia o modelo co elemento raíz do documento XML.

O contido dentro do elemento **<xsl:template>** define o contido HTML que se escribirá na saída.

As dúas derradeiras liñas definen o fin do modelo e o fin da folla de estilos.

O resultado do exemplo anterior é un pouco frustrante xa que non copiamos ningún dato dende o arquivo XML á saída. No seguinte apartado veremos como empregar o elemento **xsl:value-of** para seleccionar valores dos elementos XML.

1.7 Elemento <xsl:value-of>

O elemento **<xsl:value-of>** emprégase para extraer os valores dun nodo determinado do arquivo XML, e enviar eses valores á saída transformados:

Exemplo:

```
<?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>
```

Nota: O atributo **select** do exemplo anterior, contén unha expresión XPath. Esa expresión XPath selecciona os elementos title e artist respectivamente.

Pero no caso anterior soamente seleccionou o primeiro elemento title e artist do arquivo XML. No seguinte apartado veremos como recorrer todo o arquivo XML e amosar todos os rexistros que faltan empregando o elemento **<xsl:for-each>**.

1.8 Elemento <xsl:for-each>

O elemento **<xsl:for-each>** permítenos facer bucles no XSLT. Este elemento pode ser usado para seleccionar cada un dos elementos dentro dun conxunto especificado.

```
<?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>
```

```

        </table>
    </body>
</html>
</xsl:template>

</xsl:stylesheet>

```

Nota: O valor do atributo select é unha expresión XPath. A expresión XPath funciona como un sistema típico de arquivos dun sistema operativo.

1.8.1 Filtrando a saída

Podemos filtrar a saída dende o arquivo XML engadindo un criterio ó atributo select no elemento <xsl:for-each>.

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

Operadores dispoñibles nos filtros:

- = igual
- != distinto
- < menor que
- > maior que

Exemplo anterior cun filtro:

```

<?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[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>

```

1.9 Elemento <xsl:sort>

O elemento <xsl:sort> emprégase para ordenar a saída.

Simplemente teremos que engadir o elemento **<xsl:sort>** dentro de cada elemento <xsl:for-each> no arquivo 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">

```

```

        <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>

```

Nota: O atributo **select** indica que elemento XML temos que ordenar.

1.10 Elemento <xsl:if>

O elemento <xsl:if> emprégase para crear unha condición a avaliar no contido XML.

A sintaxe é a seguinte:

```

<xsl:if test="expresion">
  ... saída cando a expresión é verdadeira ...
</xsl:if>

```

Para engadir unha condición, teremos que engadir o elemento <xsl:if> dentro do elemento <xsl:for-each> no arquivo XSL.

Exemplo:

```

<?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 > 10">
            <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>

```

Nota: O valor do atributo obrigatorio **test** contén a expresión que queremos avaliar.

O código anterior amosará o elemento título e artista dos CD que teñen un prezo maior que 10.

1.11 Elemento <xsl:choose>

O elemento <xsl:choose> emprégase xunto con <xsl:when> e <xsl:otherwise> para expresar condicións múltiples.

Sintaxe:

```

<xsl:choose>

```

```

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

```

Para avaliar condicións múltiples no arquivo XML, engadiremos os elementos <xsl:choose>, <xsl:when> e <xsl:otherwise> no arquivo 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>
        <xsl:choose>
          <xsl:when test="price > 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>

```

O código anterior engadirá unha cor rosa de fondo á columna do Artista cando o prezo do CD sexa maior que 10.

Outro exemplo:

```

<?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 > 10">
            <td bgcolor="#ff00ff">
              <xsl:value-of select="artist"/></td>
            </xsl:when>
            <xsl:when test="price > 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>

```

```

        </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>

```

O código anterior engadirá unha color rosa de fondo á columna do Artista cando o prezo do CD é maior que 10 e unha cor gris de fondo cando o prezo do CD é maior que 9 e menor ou igual a 10.

1.12 Elemento <xsl:apply-templates>

O elemento <xsl:apply-templates> aplica un modelo ao elemento actual ou aos nodos elementos fillos do actual.

Se engadimos un atributo select ao elemento <xsl:apply-templates> procesará soamente o elemento fillo que coincida co valor do atributo. Podemos usar un atributo select para especificar a orde na cal serán procesados os nodos fillos.

Observa o seguinte exemplo:

```

<?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>

```