

1. A3. Transformacións XSLT avanzadas

1.1 Introducción

Na actividade que nos ocupa aprenderanse os seguintes conceptos e manexo de destrezas:

- Definir e empregar especificacións de transformacións XSLT que inclúan elementos como:
 - Dous ou máis patróns aplicados sobre o mesmo nodo do documento orixe.
 - Variables e estruturas de control e iteración.

1.2 Actividade

Aplicar varios patróns ao mesmo nodo

Nun documento XSLT non se deben crear varios patróns que fagan referencia ao mesmo nodo do documento orixe. Cando ao procesar un nodo do documento orixe, o procesador XSLT atopa dous ou máis patróns cun atributo "match" que lle corresponda, o seu comportamento pode ser diferente dependendo do procesador XSLT que esteamos a empregar. Pode:

- Xerar unha mensaxe de erro.
- Aplicar o patrón que apareza máis tarde no documento XSLT.

Por exemplo, se sobre o documento XML:

```
<?xml version="1.0" encoding="utf-8"?>
<módulo>
  <profesor>Xaime Louzán</profesor>
</módulo>
```

Aplicáramos a seguinte transformación:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/módulo/profesor">
    <patrón num="1" />
  </xsl:template>
  <xsl:template match="/módulo/profesor">
    <patrón num="2" />
  </xsl:template>
</xsl:stylesheet>
```

Poderíamos obter como resultado:

```
<?xml version="1.0" encoding="UTF-8"?>
  <patrón num="2"/>
```

Pero se o que queremos é transformar un ou varios nodos de dúas ou máis formas diferentes, necesitamos que o procesador aplique máis de un patrón sobre o mesmo nodo.

Para lograr isto, empregamos o atributo "mode". Este atributo aplícase aos elementos

"<xsl:template>" de xeito que cando varios patróns se aplican ao mesmo nodo, os valores dos seus atributos "mode" deben ser distintos.

Un patrón cun atributo "mode" concreto execútase cando é chamado empregando un elemento "<xsl:apply-templates>" con ese mesmo atributo "mode". Por exemplo, supoñamos que partindo do seguinte documento XML:

```
<?xml version="1.0" encoding="utf-8"?>
<ciclo siglas="ASIR">
  <módulo>Linguaxes de Marcas</módulo>
  <módulo>Servizos de Rede</módulo>
  <módulo>Fundamentos Hardware</módulo>
</ciclo>
```

Queremos obter unha páxina web como a seguinte, onde o texto correspondente a cada un dos módulos do documento orixinal transfórmase primeiro nun encabezado, e logo nunha lista.



A forma de facelo é cun documento XSLT como o seguinte:

```
<?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></head>
      <body>
        <xsl:apply-templates mode="encabezados"/>
        <ul>
          <xsl:apply-templates mode="lista"/>
        </ul>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="módulo" mode="encabezados">
    <h2><xsl:value-of select="text()" /></h2>
  </xsl:template>
  <xsl:template match="módulo" mode="lista">
    <li><xsl:value-of select="text()" /></li>
  </xsl:template>
</xsl:stylesheet>
```

Prioridade dos patróns

En realidade, existe un sistema de prioridades que indica ao procesador XSLT cal é o patrón que debe coller para procesar un nodo do documento orixe, en caso de que teña que escoller entre varios. O procesador XSLT escollerá sempre o patrón con maior prioridade; no caso de non atopar un único patrón con maior prioridade cos outros, é cando escollerá o último que apareza ou xerará unha mensaxe de erro.

O sistema de prioridades pode ser explícito (nos mesmos indicamos cal é a prioridade de cada patrón) ou implícito. Cando queremos indicar nos mesmos a prioridade dun patrón, o faremos engadíndolle o atributo "priority".

```
<xsl:template match="módulo" priority="2">
...
</xsl:template>
```

As prioridades implícitas entran en xogo cando creamos un patrón sen atributo "priority". Neste caso, para poder escoller o procesador XSLT asígnalles a estes patróns unha prioridade implícita entre -0,5 e +0,5, que será maior canto máis específica sexa a expresión do seu atributo "match".

Por exemplo, se transformamos o documento XML.

```
<?xml version="1.0" encoding="utf-8"?>
<módulo>
  <profesor>Xaime Louzán</profesor>
</módulo>
```

Empregando o seguinte patrón.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/módulo">
    <xsl:apply-templates />
  </xsl:template>
  <xsl:template match="profesor">
    <patrón num="1" />
  </xsl:template>
  <xsl:template match="*">
    <patrón num="2" />
  </xsl:template>
</xsl:stylesheet>
```

O nodo "<profesor>" transformarase empregando o patrón "<xsl:template match='profesor'>", que aínda que aparece antes no documento XSLT, é máis específico que o patrón "<xsl:template match='*'>", e por tanto ten maior prioridade implícita. Obteremos como resultado de saída.

```
<?xml version="1.0" encoding="UTF-8"?>
<patrón num="1"/>
```

Copiar nodos do documento orixe ao documento de saída

Nos patróns de XSLT temos dúas opcións para copiar nodos presentes no documento orixe ao documento de saída:

- "<xsl:copy>" realiza unha copia sinxela do elemento ao que se refire o patrón. Non copia os seus atributos, o seu texto ou os seus fillos. Soamente o elemento. Este tipo de co-

pia recibe o nome de copia superficial (*shallow copy*). Se quixéramos procesar tamén o seu contido, habería que facelo de forma específica.

- "`<xsl:copy-of>`" polo contrario fai unha copia profunda (*deep copy*) do elemento e traslada ao documento resultante tamén o texto que contén, os seus fillos e atributos. A diferenza do anterior, este elemento debe estar baleiro e é obrigatorio indicar cun atributo "`select`" a expresión XPath correspondente ao nodo ou nodos que queremos copiar.

Así, seguindo co noso documento XML de exemplo.

```
<?xml version="1.0" encoding="utf-8"?>
<módulo>
  <profesor>Xaime Louzán </profesor>
</módulo>
```

O resultado obtido por un patrón que empregue "`<xsl:copy>`" como o seguinte.

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

Será:

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

E cando empregamos no patrón "`<xsl:copy-of>`".

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/módulo">
    <xsl:copy-of select="." />
  </xsl:template>
</xsl:stylesheet>
```

O resultado pasa a ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<módulo>
  <profesor>Xaime Louzán</profesor>
</módulo>
```



Na tarefa 1 traballaremos con transformacións que apliquen varios patróns a un mesmo elemento do documento orixe.

Variables

Ao igual que noutros linguaxes de programación, en XSLT podemos definir e empregar variables. O elemento "`<xsl:variable>`" define unha variable. O nome da variable establécse-

se empregando o atributo obrigatorio "name="nome da variable", e o seu valor pódese obter a partires de:

- unha expresión XPath empregando o atributo "select".

```
<xsl:variable name="num_profesores" select="count(profesor)" />
```

- o contido do elemento "<xsl:variable>".

```
<xsl:variable name="var">valor da variable</xsl:variable>
<!-- Tamén podería poñerse o valor nunha cadea de texto dentro do atributo
select -->
<xsl:variable name="var" select="'valor da variable'" />
```

É importante destacar que o valor dunha variable en XSLT non se pode cambiar. Asígnase-lle cando se declara e mantense ese mesmo valor de aí en adiante.

As variables pódense definir no interior dun patrón ou fora deles, como fillo directo do elemento "<xsl:stylesheet>". Segundo sexa o caso, teremos:

- unha variable local a un patrón (non existirá fora dese patrón).
- unha variable global ao documento XSLT (poderase empregar dende calquera patrón do documento).

Entre as utilidades que podemos darlles ás variables en XSLT están:

- Definir valores que poidan cambiarse de xeito doado para alterar o comportamento da transformación.

```
<xsl:variable name="cor_fondo">#dedede</xsl:variable>
```

- Almacenar valores de conxuntos de nodos para empregalos posteriormente.

```
<xsl:variable name="encabezado">
  <tr>
    <th>Nome</th>
    <th>Descrición</th>
  </tr>
</xsl:variable>
```

- Almacenar expresións complexas, para aumentar a lexibilidade do código e o rendemento do procesador XSLT.

```
<xsl:variable name="pelis"
  select="//pelicula[actúa/@id="//actor[nome='Elisabeth Shue']/@id]" />
```

- Almacenar o valor devolto por elementos XSLT.

```
<xsl:variable name="encabezado">
  <xsl:element name="tr">
    <xsl:element name="th">Nome</xsl:element>
    <xsl:element name="th">Descrición</xsl:element>
  </xsl:element>
</xsl:variable>
```

Empregar variables

Para obter o valor dunha variable, antepónse ao seu nome o símbolo "\$". Ao executar a transformación, a expresión "\$variable" substituirase polo valor da variable. Por exemplo:

```

<xsl:variable name="cor_fondo">#dedede</xsl:variable>
<xsl:element name="body">
  <xsl:attribute name="bgcolor">
    <xsl:value-of select="$cor_fondo" />
  </xsl:attribute>
</xsl:element>

```

Se queremos obter o valor dunha variable dentro dun atributo, tamén podemos empregar chaves:

```

<xsl:variable name="cor_fondo">#dedede</xsl:variable>
<body bgcolor="{ $cor_fondo}" />

```

Cando a variable contén un conxunto de nodos, poderíamos copialos ao documento de saída empregando "<xsl:copy-of>".

```

<xsl:variable name="pelis"
  select="//película[actúa/@id=//actor[nome='Elisabeth Shue']/@id]" />
<xsl:copy-of select="$pelis" />

```

Estruturas de control

Como parte dos patróns, podemos usar certas estruturas de control no procesamento dos elementos do documento orixinal.

Condições simples

O elemento "<xsl:if>" permítenos procesar unha parte do patrón unicamente cando se cumpre unha condición. A condición especificase mediante unha expresión dentro do atributo obrigatorio "test". Cando o resultado da expresión é certo, execútanse o contido do elemento.

Por exemplo, partindo do seguinte documento XML.

```

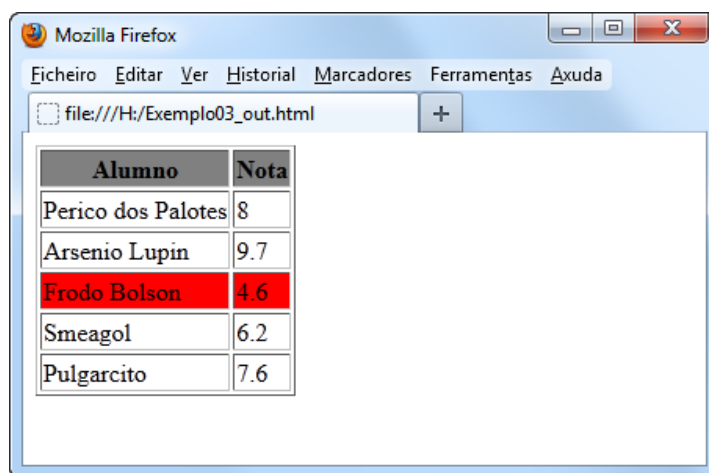
<?xml version="1.0" encoding="UTF-8"?>
<alumnos>
  <alumno>
    <nome>Perico dos Palotes</nome>
    <nota>8</nota>
  </alumno>
  <alumno>
    <nome>Arsenio Lupin</nome>
    <nota>9.7</nota>
  </alumno>
  <alumno>
    <nome>Frodo Bolson</nome>
    <nota>4.6</nota>
  </alumno>
  <alumno>
    <nome>Smeagol</nome>
    <nota>6.2</nota>
  </alumno>
  <alumno>
    <nome>Pulgarcito</nome>
    <nota>7.6</nota>
  </alumno>
</alumnos>

```

Poderíamos transformalo nunha táboa HTML na que as filas correspondentes a alumnos con nota < 5 teñan o fondo vermello.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="UTF-8" indent="yes" method="html"/>
  <xsl:template match="/">
    <html>
      <head></head>
      <body>
        <table border="1">
          <tr bgcolor="gray">
            <th>Alumno</th>
            <th>Nota</th>
          </tr>
          <xsl:apply-templates select="alumnos/alumno" />
        </table>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="alumno">
    <xsl:element name="tr">
      <xsl:if test="nota < 5">
        <xsl:attribute name="bgcolor">red</xsl:attribute>
      </xsl:if>
      <td><xsl:value-of select="nome" /></td>
      <td><xsl:value-of select="nota" /></td>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

O resultado obtido sería:



Alumno	Nota
Perico dos Palotes	8
Arsenio Lupin	9.7
Frodo Bolson	4.6
Smeagol	6.2
Pulgarcito	7.6

Fíxate que para evitar problemas cos caracteres de apertura de elementos, os operadores "<" e ">" deben substituírse respectivamente por "<" e ">".

Condições múltiples

O elemento "<xsl:if>" permítenos avaliar o resultado dunha expresión booleana. Cando queremos crear unha condición máis complexa, na que o resultado da expresión poda ser comparado con varios posibles valores, teremos que empregar o elemento "<xsl:choose>".

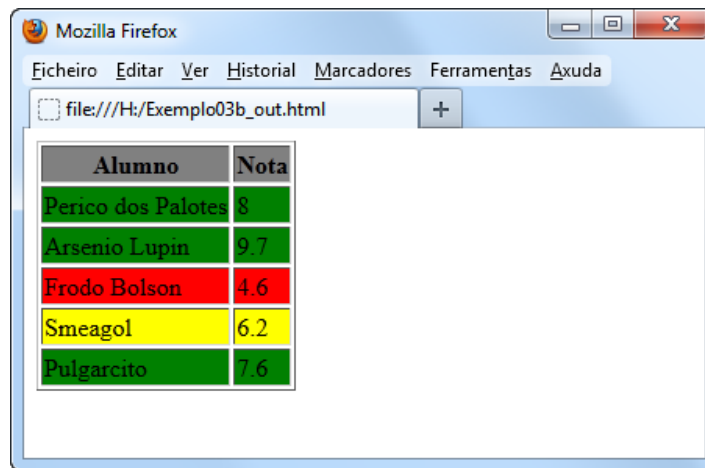
Dentro do elemento "<xsl:choose>" aníñanse tantos elementos "<xsl:when>" como se-
xa necesario, cadanseu co seu respectivo atributo "test". Cando a expresión correspondente
a un elemento "<xsl:when>" é certa, procésase o seu contido e detense o procesamento do
resto.

Se non se cumpre a condición asociada a ningún dos elementos "<xsl:when>", procesa-
rase o contido do elemento "<xsl:otherwise>" en caso de que exista.

Por exemplo, para transformar o mesmo documento anterior poñendo unha cor distinta
para cada nota poderíamos facer:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="UTF-8" indent="yes" method="html"/>
  <xsl:template match="/">
    <html>
      <head></head>
      <body>
        <table border="1">
          <tr bgcolor="gray">
            <th>Alumno</th>
            <th>Nota</th>
          </tr>
          <xsl:apply-templates select="alumnos/alumno" />
        </table>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="alumno">
    <xsl:element name="tr">
      <xsl:choose>
        <xsl:when test="nota < 5">
          <xsl:attribute name="bgcolor">red</xsl:attribute>
        </xsl:when>
        <xsl:when test="nota < 7">
          <xsl:attribute name="bgcolor">yellow</xsl:attribute>
        </xsl:when>
        <xsl:otherwise>
          <xsl:attribute name="bgcolor">green</xsl:attribute>
        </xsl:otherwise>
      </xsl:choose>
      <td><xsl:value-of select="nome" /></td>
      <td><xsl:value-of select="nota" /></td>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

E obteríamos o seguinte resultado:



Estruturas de iteración

Ata o de agora estivemos a empregar patróns para procesar os conxuntos de nodos contidos nun elemento. Pero en XSLT existe outro xeito de facelo: empregando o elemento "`<xsl:for-each>`".

O elemento "`<xsl:for-each>`" debe ir acompañado dun atributo "select" que faga referencia a un conxunto de nodos. O contido do elemento "`<xsl:for-each>`" procesarase unha vez por cada nodo no conxunto de nodos referenciado.

Por exemplo, poderíamos refacer a transformación anterior procesando os alumnos de forma iterativa do seguinte xeito.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="UTF-8" indent="yes" method="html"/>
  <xsl:template match="/">
    <html>
      <head></head>
      <body>
        <table border="1">
          <tr bgcolor="gray">
            <th>Alumno</th>
            <th>Nota</th>
          </tr>
          <xsl:for-each select="alumnos/alumno">
            <xsl:element name="tr">
              <xsl:choose>
                <xsl:when test="nota < 5">
                  <xsl:attribute
name="bgcolor">red</xsl:attribute>
                </xsl:when>
                <xsl:when test="nota < 7">
                  <xsl:attribute
name="bgcolor">yellow</xsl:attribute>
                </xsl:when>
                <xsl:otherwise>
                  <xsl:attribute
name="bgcolor">green</xsl:attribute>
                </xsl:otherwise>
              </xsl:choose>
              <td><xsl:value-of select="nome" /></td>
              <td><xsl:value-of select="nota" /></td>
            </xsl:element>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </template>
</xsl:stylesheet>
```

```

        </xsl:for-each>
    </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

Especificar unha orde para os nodos

Como acabamos de ver, temos dúas formas para procesar un conxunto de nodos: empregando un patrón cun atributo "match" ou empregando un elemento "<xsl:for-each>" cun atributo "select".

Empregando calquera destes dous métodos, o procesador XSLT escolle os nodos do conxunto de nodos para procesalos na mesma orde na que aparecen no documento orixinal. No exemplo anterior, procesará aos alumnos tal e como aparecen no documento XML.

En XSLT podemos cambiar a orde na que se procesan os nodos dun conxunto de nodos engadindo, con calquera das dúas formas indicadas, o elemento "<xsl:sort>". Este elemento debe aparecer dentro dun "<xsl:apply-templates>" ou xusto a continuación dun "<xsl:for-each>", sen outros elementos intermedios.

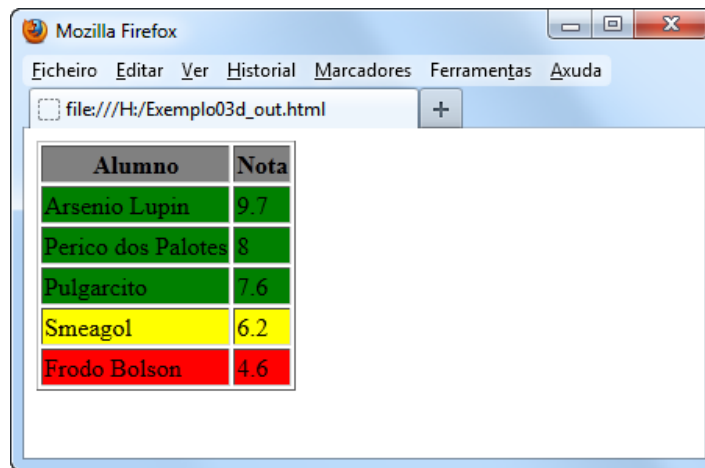
O elemento "<xsl:sort>" ten unha serie de atributos opcionais que indican o tipo de ordenamento a levar a cabo:

- "select" indica a chave de ordenación. Se non figura este atributo, suponse como chave de ordenación "select=". "
- "order" pode tomar dous valores para indicar se queremos facer ordenación ascendente ("ascending", que é o valor por defecto) ou ordenación descendente ("descending").
- "case-order" tamén pode tomar dous valores ("upper-first" ou "lower-first") en función de se queremos que as letras maiúsculas se ordenen antes ou despois das minúsculas.
- "data-type" indica o tipo dos datos que imos a ordenar. Por defecto trátanse como texto ("text") pero ás veces é útil tratalos como números ("number"), de xeito por exemplo que "7" vaia antes de "12".

Por exemplo, no exemplo anterior para ordenar os nodos pola nota de xeito descendente teríamos que engadir despois de "<xsl:for-each select='alumnos/alumno'>" un elemento:

```
<xsl:sort select="nota" order="descending" data-type="number" />
```

E obteríamos:



A screenshot of a Mozilla Firefox browser window. The address bar shows a local file path: file:///H:/Exemplo03d_out.html. The main content area displays a table with two columns: 'Alumno' and 'Nota'. The table contains six rows of data, each with a different background color for the 'Alumno' cell.

Alumno	Nota
Arsenio Lupin	9.7
Perico dos Palotes	8
Pulgarcito	7.6
Smeagol	6.2
Frodo Bolson	4.6

É posible engadir máis de un criterio de ordenación, de xeito que os nodos que teñan a mesma orde segundo o primeiro criterio pase a ordenarse seguindo o segundo.

Por exemplo, para ordenar polo nome a aqueles alumnos coa mesma nota (supoñendo que houbera algún), teríamos que facer:

```
<xsl:sort select="nota" order="descending" data-type="number" />
<xsl:sort select="nome" />
```



Agora imos facer a tarefa 2, na que traballaremos con algúns destes conceptos.

1.3 Tarefas

As tarefas propostas son as seguintes.

- **Tarefa 1. Aplicar varios patróns a un mesmo elemento.** Nesta tarefa teremos que crear transformacións que procesen un mesmo elemento do documento orixe empregando dous ou máis patróns distintos.
- **Tarefa 2. Transformacións con variables, decisións e estruturas iterativas.** Nesta tarefa veremos transformacións que apliquen patróns con estruturas de control e iterativas.

1.3.1 Tarefa 1. Aplicar varios patróns a un mesmo elemento

Tomando como base o seguinte documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<equipos>
  <máquina nome="PC017">
    <hardware>
      <tipo>PC Sobremesa</tipo>
      <fabricante>Dell</fabricante>
      <procesador marca="Intel" num_nucleos="4" velocidade="3,1">i7</procesador>
      <memoria tecnoloxía="DDR3">8</memoria>
      <disco tecnoloxía="SATA" capacidade="2000"/>
      <gravadora tipo="DVD"/>
    </hardware>
    <config>
      <OS>Windows 7</OS>
      <IP>192.168.20.105</IP>
      <gateway>192.168.20.1</gateway>
    </config>
  </máquina>
  <máquina nome="GALILEO">
    <hardware>
      <tipo>Torre</tipo>
      <fabricante>Fujitsu-Siemens</fabricante>
      <procesador marca="Intel" num_nucleos="4" velocidade="3">Xeon</procesador>
      <memoria tecnoloxía="DDR2">2</memoria>
      <disco tecnoloxía="SCSI" capacidade="200"/>
      <disco tecnoloxía="SCSI" capacidade="200"/>
      <disco tecnoloxía="SCSI" capacidade="200"/>
      <lectora tipo="DVD"/>
    </hardware>
    <config>
      <role>Servidor de dominio</role>
      <OS>Windows 2008 Server R2</OS>
      <IP>192.168.20.10</IP>
      <gateway>192.168.20.1</gateway>
    </config>
  </máquina>
</equipos>
```

a) Tarefa 1_a

Obter un listado en formato XML dos discos e das memorias que figuran no documento XML orixe como o seguinte:

```

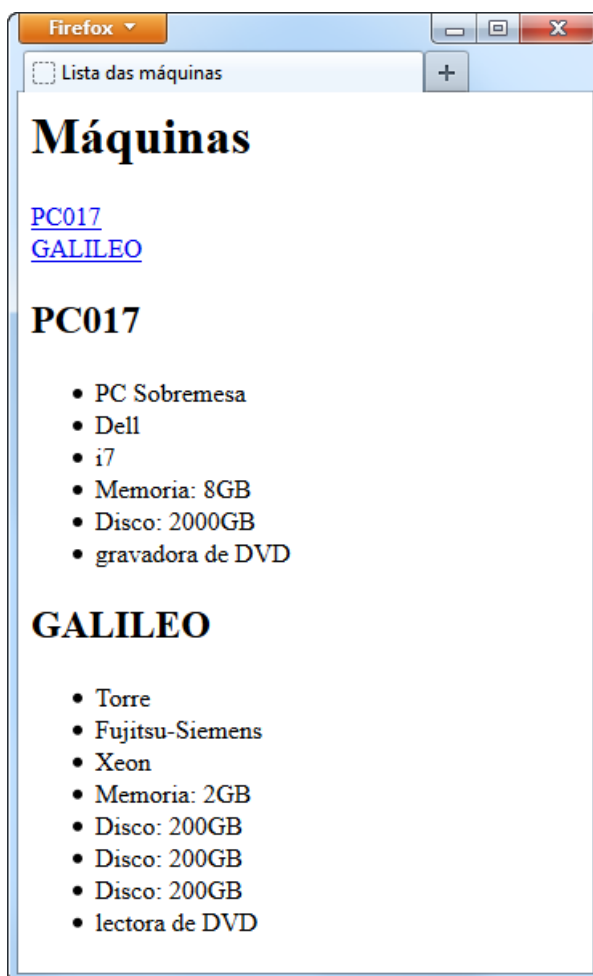
<?xml version="1.0" encoding="UTF-8"?>
<almacenamento>
  <discos num="4">
    <disco tecnoloxía="SATA" capacidade="2000"/>
    <disco tecnoloxía="SCSI" capacidade="200"/>
    <disco tecnoloxía="SCSI" capacidade="200"/>
    <disco tecnoloxía="SCSI" capacidade="200"/>
  </discos>
  <memorias num="2">
    <memoria tecnoloxía="DDR3">8</memoria>
    <memoria tecnoloxía="DDR2">2</memoria>
  </memorias>
</almacenamento>

```

b) Tarefa 1_b

Obter unha páxina HTML na que figure primeiro un índice cos nomes das máquinas, e logo unha lista non numerada na que se detallan as características hardware de cada máquina.

O obxectivo é xerar algo como o seguinte, tendo en conta que ao pinchar co rato no enlace de cada máquina o navegador amosará as características correspondentes a esa máquina:



Autoavaliación

a) Tarefa 1_a

Unha posible solución á transformación é:

```

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

```

```

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="UTF-8" indent="yes" method="xml"/>
  <xsl:template match="/">
    <xsl:element name="almacenamento">
      <xsl:element name="discos">
        <xsl:attribute name="num">
          <xsl:value-of select="count(//disco)" />
        </xsl:attribute>
        <xsl:copy-of select="equipos/máquina/hardware/disco" />
      </xsl:element>
      <xsl:element name="memorias">
        <xsl:attribute name="num">
          <xsl:value-of select="count(//memoria)" />
        </xsl:attribute>
        <xsl:copy-of select="equipos/máquina/hardware/memoria" />
      </xsl:element>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>

```

Outra solución empregando o atributo "mode" sería:

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="UTF-8" indent="yes" method="xml"/>
  <xsl:template match="/">
    <almacenamento>
      <discos num="{count(equipos/máquina/hardware/disco)}">
        <xsl:apply-templates select="equipos/máquina" mode="discos"/>
      </discos>
      <memorias num="{count(equipos/máquina/hardware/memoria)}">
        <xsl:apply-templates select="equipos/máquina" mode="memorias"/>
      </memorias>
    </almacenamento>
  </xsl:template>
  <xsl:template match="máquina" mode="discos">
    <xsl:copy-of select="hardware/disco" />
  </xsl:template>
  <xsl:template match="máquina" mode="memorias">
    <xsl:copy-of select="hardware/memoria" />
  </xsl:template>
</xsl:stylesheet>

```

b) Tarefa 1_b

Unha forma de acadar o obxectivo é empregando o seguinte documento XSLT:

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="UTF-8" method="html"/>
  <xsl:template match="/">
    <html>
      <head>
        <title>Lista das máquinas</title>
      </head>
      <body>
        <h1>Máquinas</h1>
        <xsl:apply-templates select="equipos/máquina" mode="enlaces" />
      </body>
    </html>
  </xsl:template>
  <xsl:template match="máquina" mode="enlaces">
    <a href="{hardware/disco}">{hardware/disco}</a>
  </xsl:template>
</xsl:stylesheet>

```

```

        <xsl:apply-templates select="equipos/máquina" mode="detalle" />
    </body>
</html>
</xsl:template>
<xsl:template match="máquina" mode="enlaces">
    <a href="#{@nome}"><xsl:value-of select="@nome" /></a><br />
</xsl:template>
<xsl:template match="máquina" mode="detalle">
    <a name="{@nome}"><h2><xsl:value-of select="@nome" /></h2></a>
    <ul>
        <xsl:apply-templates select="hardware/*" />
    </ul>
</xsl:template>
<xsl:template match="memoria">
    <li>Memoria: <xsl:value-of select="text()" />GB</li>
</xsl:template>
<xsl:template match="disco">
    <li>Disco: <xsl:value-of select="@capacidade" />GB</li>
</xsl:template>
<xsl:template match="lectora|gravadora">
    <li><xsl:value-of select="name()" /> de <xsl:value-of select="@tipo"
/></li>
</xsl:template>
<xsl:template match="*">
    <li><xsl:value-of select="text()" /></li>
</xsl:template>
</xsl:stylesheet>

```

1.3.2 Tarefa 2. Transformacións con variables, decisións e estruturas iterativas

Tomando como base o seguinte documento XML:

```

<?xml version="1.0" encoding="utf-8"?>
<equipos>
  <máquina nome="PC017">
    <hardware>
      <tipo>PC Sobremesa</tipo>
      <fabricante>Dell</fabricante>
      <procesador marca="Intel" num_nucleos="4" velocidade="3.1">i7</procesador>
      <memoria tecnoloxía="DDR3">8</memoria>
      <disco tecnoloxía="SATA" capacidade="2000"/>
      <gravadora tipo="DVD"/>
    </hardware>
    <config>
      <OS>Windows 7</OS>
      <IP>192.168.10.105</IP>
      <gateway>192.168.20.1</gateway>
    </config>
  </máquina>
  <máquina nome="PC053">
    <hardware>
      <tipo>Semitorre</tipo>
      <memoria>0.5</memoria>
      <disco capacidade="40"/>
      <lectora tipo="CD"/>
    </hardware>
    <config>
      <OS>Windows XP</OS>
    </config>
  </máquina>
</equipos>

```

```

    </config>
</máquina>
<máquina nome="PC007">
  <hardware>
    <tipo>Semitorre</tipo>
    <memoria tecnologia="DDR">0.5</memoria>
    <disco capacidade="40"/>
    <lectora tipo="CD"/>
  </hardware>
  <config>
    <OS>Windows XP</OS>
  </config>
  <notas>Sin tarxeta de rede</notas>
</máquina>
<máquina nome="PR003">
  <hardware>
    <tipo>Impresora Inxección</tipo>
    <fabricante>Lexmark</fabricante>
  </hardware>
  <config/>
</máquina>
<máquina nome="PC011">
  <hardware>
    <tipo>Semitorre</tipo>
    <memoria>1</memoria>
    <disco capacidade="80"/>
    <lectora tipo="CD"/>
  </hardware>
  <config>
    <OS>Windows 2000 SP4</OS>
    <IP>192.168.10.221</IP>
  </config>
</máquina>
<máquina nome="PC019">
  <hardware>
    <tipo>Semitorre</tipo>
    <procesador marca="AMD" velocidade="1.4">Athlon</procesador>
    <memoria>0.5</memoria>
    <disco capacidade="40"/>
    <gravadora tipo="CD"/>
  </hardware>
  <config>
    <OS>Mandriva 2007</OS>
    <IP>192.168.10.45</IP>
    <gateway>192.168.10.1</gateway>
  </config>
</máquina>
<máquina nome="PR007">
  <hardware>
    <tipo>Impresora Láser</tipo>
    <fabricante>OKI</fabricante>
  </hardware>
  <config/>
  <notas>Monocromo, dúplex, rede</notas>
</máquina>
<máquina nome="COPERNICO">
  <hardware>
    <tipo>Torre</tipo>
    <fabricante>Fujitsu-Siemens</fabricante>
    <procesador marca="Intel" num_nucleos="4" velocidade="3">Xeon</procesador>

```



```

    <memoria tecnologia="DDR">2</memoria>
    <disco tecnologia="SCSI" capacidade="500"/>
    <disco tecnologia="SCSI" capacidade="500"/>
    <gravadora tipo="DVD"/>
  </hardware>
  <config>
    <role>Servidor de dominio</role>
    <OS>Windows 2003 Server R2</OS>
    <IP>192.168.200.11</IP>
    <gateway>192.168.20.1</gateway>
  </config>
</maquina>
<maquina nome="GALILEO">
  <hardware>
    <tipo>Torre</tipo>
    <fabricante>Fujitsu-Siemens</fabricante>
    <procesador marca="Intel" num_nucleos="4" velocidade=
de="3">Xeon</procesador>
    <memoria tecnologia="DDR2">2</memoria>
    <disco tecnologia="SCSI" capacidade="200"/>
    <disco tecnologia="SCSI" capacidade="200"/>
    <disco tecnologia="SCSI" capacidade="200"/>
    <lectora tipo="DVD"/>
  </hardware>
  <config>
    <role>Servidor de dominio</role>
    <OS>Windows 2008 Server R2</OS>
    <IP>192.168.20.10</IP>
    <gateway>192.168.20.1</gateway>
  </config>
</maquina>
<maquina nome="KEPLER">
  <hardware>
    <tipo>Rack</tipo>
    <fabricante>HP</fabricante>
    <procesador marca="Intel" num_nucleos="2" velocidade="3">Core2
Duo</procesador>
    <memoria tecnologia="DDR2">4</memoria>
    <disco tecnologia="SATA" capacidade="500"/>
    <disco tecnologia="SATA" capacidade="500"/>
    <disco tecnologia="SATA" capacidade="500"/>
    <gravadora tipo="DVD"/>
  </hardware>
  <config>
    <role>Servidor de arquivos</role>
    <OS>Ubuntu 8.04 Server</OS>
    <IP>192.168.10.10</IP>
    <gateway>192.168.10.1</gateway>
  </config>
</maquina>
<maquina nome="NEWTON">
  <hardware>
    <tipo>Rack</tipo>
    <fabricante>HP</fabricante>
    <procesador marca="Intel" num_nucleos="2" velocidade="3">Core2
Duo</procesador>
    <memoria tecnologia="DDR2">4</memoria>
    <disco tecnologia="SATA" capacidade="500"/>
    <disco tecnologia="SATA" capacidade="500"/>
    <gravadora tipo="DVD"/>
  </hardware>

```

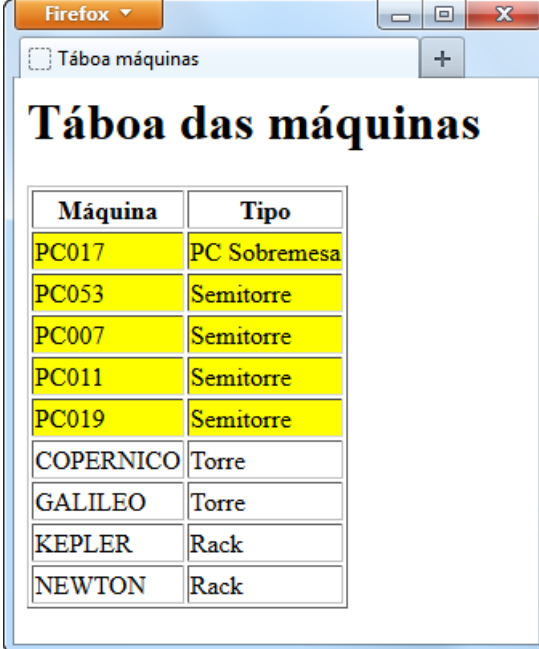
```

<config>
  <role>Servidor web</role>
  <OS>Ubuntu 8.04 Server</OS>
  <IP>192.168.10.11</IP>
  <gateway>192.168.10.1</gateway>
</config>
</máquina>
</equipos>

```

a) Tarefa 2_a

Xerar como saída da transformación un documento HTML que conteña unha táboa como a seguinte.



Máquina	Tipo
PC017	PC Sobremesa
PC053	Semitorre
PC007	Semitorre
PC011	Semitorre
PC019	Semitorre
COPERNICO	Torre
GALILEO	Torre
KEPLER	Rack
NEWTON	Rack

Na táboa deben figurar todas as máquinas, salvo aquelas de tipo "Impresora" (nas que o tipo comece con "Impresora"). Ademais deberán marcarse con fondo amarelo aquelas máquinas de tipo "PC Sobremesa" ou "Semitorre".

Facer a solución de dúas formas: primeiro empregando patróns, e a continuación con estruturas iterativas.

b) Tarefa 2_b

Modificar as transformacións obtidas na tarefa anterior, para que a táboa amose os elementos ordenados polo seu tipo, e dentro dos elementos dun mesmo tipo polo seu nome.

c) Tarefa 2_c

Xerar como saída da transformación un documento HTML que conteña unha táboa como a seguinte.

Máquina	Tipo	OS	Capacidade HD
PC017	PC Sobremesa	Windows 7	2000 GB
KEPLER	Rack	Ubuntu 8.04 Server	1500 GB
COPERNICO	Torre	Windows 2003 Server R2	1000 GB
NEWTON	Rack	Ubuntu 8.04 Server	1000 GB
GALILEO	Torre	Windows 2008 Server R2	600 GB
PC011	Semitorre	Windows 2000 SP4	80 GB
PC053	Semitorre	Windows XP	40 GB
PC007	Semitorre	Windows XP	40 GB
PC019	Semitorre	Mandriva 2007	40 GB

Na táboa soamente deberán figurar aquelas máquinas con sistema operativo (nas que exista o elemento "os", marcando en amarelo as filas relativas a máquinas con sistema operativo da familia "Windows", e ordenándoas de maior a menor capacidade total dos seus discos duros.

Por último, a cor correspondente ao texto da capacidade variará entre o vermello ("FF0000") cando é igual ou maior de 1000GB, o laranxa-vermello ("FF4500") cando a súa capacidade está entre os 500 e os 1000GB, e laranxa ("FFA500") cando é inferior a 500GB.

Facer a transformación de dous xeitos: empregando patróns e empregando estruturas iterativas.

Autoavaliación

a) Tarefa 2_a

Unha posible solución á transformación empregando patróns é:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="UTF-8" method="html"/>
  <xsl:template match="/">
    <html>
      <head>
        <title>Táboa máquinas</title>
      </head>
      <body>
        <h1>Táboa das máquinas</h1>
        <table border="1">
          <tr>
            <th>Máquina</th>
            <th>Tipo</th>
          </tr>
          <xsl:apply-templates select="equipos/máquina[not (starts-
with(hardware/tipo, 'Impresora'))]" />
        </table>
```

```

        </body>
    </html>
</xsl:template>
<xsl:template match="máquina">
    <xsl:element name="tr">
        <xsl:if test="hardware/tipo = 'Semitorre' or hardware/tipo = 'PC Sobre-
mesa'">
            <xsl:attribute name="bgcolor">yellow</xsl:attribute>
        </xsl:if>
        <td><xsl:value-of select="@nome" /></td>
        <td><xsl:value-of select="hardware/tipo" /></td>
    </xsl:element>
</xsl:template>
</xsl:stylesheet>

```

Outra versión con estruturas iterativas sería:

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output encoding="UTF-8" method="html"/>
    <xsl:template match="/">
        <html>
            <head>
                <title>Táboa máquinas</title>
            </head>
            <body>
                <h1>Táboa das máquinas</h1>
                <table border="1">
                    <tr>
                        <th>Máquina</th>
                        <th>Tipo</th>
                    </tr>
                    <xsl:for-each select="equipos/máquina[not(starts-with(hardware/tipo,
'Impresora'))]">
                        <xsl:element name="tr">
                            <xsl:if test="hardware/tipo = 'Semitorre' or hardware/tipo = 'PC So-
bre-mesa'">
                                <xsl:attribute name="bgcolor">yellow</xsl:attribute>
                            </xsl:if>
                            <td><xsl:value-of select="@nome" /></td>
                            <td><xsl:value-of select="hardware/tipo" /></td>
                        </xsl:element>
                    </xsl:for-each>
                </table>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>

```

b) Tarefa 2_b

Na versión con patróns, habería que modificar a chamada engadindo os criterios de ordenación do seguinte xeito:

```

<xsl:apply-templates select="equipos/máquina[not(starts-with(hardware/tipo,
'Impresora'))]">
    <xsl:sort select="hardware/tipo" />
    <xsl:sort select="@nome" />
</xsl:apply-templates>

```

E de xeito similar, na versión con estruturas iterativas habería que facer:

```
<xsl:for-each select="equipos/máquina[not (starts-with(hardware/tipo, 'Impre-
sora'))]">
  <xsl:sort select="hardware/tipo" />
  <xsl:sort select="@nome" />
  ...
</xsl:for-each>
```

c) Tarefa 2_c

Unha posible solución á transformación con patróns é:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="UTF-8" method="html"/>
  <xsl:template match="/">
    <html>
      <head>
        <title>Táboa de ordenadores con OS</title>
      </head>
      <body>
        <h1>Táboa de ordenadores con OS</h1>
        <table border="1">
          <tr>
            <th>Máquina</th>
            <th>Tipo</th>
            <th>OS</th>
            <th>Capacidade HD</th>
          </tr>
          <xsl:for-each select="equipos/máquina[config/OS]" >
            <xsl:sort select="sum(hardware/disco/@capacidade)" order="descending"
data-type="number" />
            <xsl:element name="tr">
              <xsl:if test="starts-with(config/OS, 'Windows')">
                <xsl:attribute name="style">background:yellow</xsl:attribute>
              </xsl:if>
              <td><xsl:value-of select="@nome" /></td>
              <td><xsl:value-of select="hardware/tipo" /></td>
              <td><xsl:value-of select="config/OS" /></td>
              <td>
                <xsl:choose>
                  <xsl:when test="sum(hardware/disco/@capacidade) >= 1000">
                    <xsl:attribute name="style">color:red</xsl:attribute>
                  </xsl:when>
                  <xsl:when test="sum(hardware/disco/@capacidade) >= 500">
                    <xsl:attribute name="style">color:#FF4500</xsl:attribute>
                  </xsl:when>
                  <xsl:otherwise>
                    <xsl:attribute name="style">color:orange</xsl:attribute>
                  </xsl:otherwise>
                </xsl:choose>
                <xsl:value-of select="sum(hardware/disco/@capacidade)" /> GB
              </td>
            </xsl:element>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
```

```
</xsl:stylesheet>
```

E a mesma solución, empregando estruturas iterativas quedaría:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="UTF-8" method="html"/>
  <xsl:template match="/">
    <html>
      <head>
        <title>Táboa de ordenadores con OS</title>
      </head>
      <body>
        <h1>Táboa de ordenadores con OS</h1>
        <table border="1">
          <tr>
            <th>Máquina</th>
            <th>Tipo</th>
            <th>OS</th>
            <th>Capacidade HD</th>
          </tr>
          <xsl:apply-templates select="equipos/máquina[config/OS]">
            <xsl:sort select="sum(hardware/disco/@capacidade)" order="descending"
data-type="number" />
          </xsl:apply-templates>
        </table>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="máquina">
    <xsl:element name="tr">
      <xsl:if test="starts-with(config/OS, 'Windows')">
        <xsl:attribute name="style">background:yellow</xsl:attribute>
      </xsl:if>
      <td><xsl:value-of select="@nome" /></td>
      <td><xsl:value-of select="hardware/tipo" /></td>
      <td><xsl:value-of select="config/OS" /></td>
      <td>
        <xsl:choose>
          <xsl:when test="sum(hardware/disco/@capacidade) >= 1000">
            <xsl:attribute name="style">color:red</xsl:attribute>
          </xsl:when>
          <xsl:when test="sum(hardware/disco/@capacidade) >= 500">
            <xsl:attribute name="style">color:#FF4500</xsl:attribute>
          </xsl:when>
          <xsl:otherwise>
            <xsl:attribute name="style">color:orange</xsl:attribute>
          </xsl:otherwise>
        </xsl:choose>
        <xsl:value-of select="sum(hardware/disco/@capacidade)" /> GB
      </td>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

2. Materiais

2.1 Textos de apoio ou de referencia

- **XML in a Nutshell, 2nd Ed**, Elliotte Rusty Harold e W. Scott Means, Ed. O'Reilly.
- **Beginning XML, 5th Ed**, Joe Fawcett, Liam R. E. Quin e Danny Ayers, Ed. Wrox.
- **Sams Teach Yourself XSLT in 21 days**, Michiel van Otegem, Ed. Sams.
- **XSLT, Mastering XML Transformations**, Doug Tidwell, Ed. O'Reilly.

2.2 Recursos didácticos

- Ordenador persoal, con navegador web e software de realización de transformacións XSLT.