

# 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.
  - Definición e utilización de patróns con nome.
  - Especificación e utilización de parámetros nas chamadas aos patróns.
  - Definición e utilización de claves e identificadores para realizar agrupamento de nodos.

## 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>" transformárase 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 copia 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 establécese 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. Asígnaselle 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 empregarlos 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="//película[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, anteponse 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.

### Condicións 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.

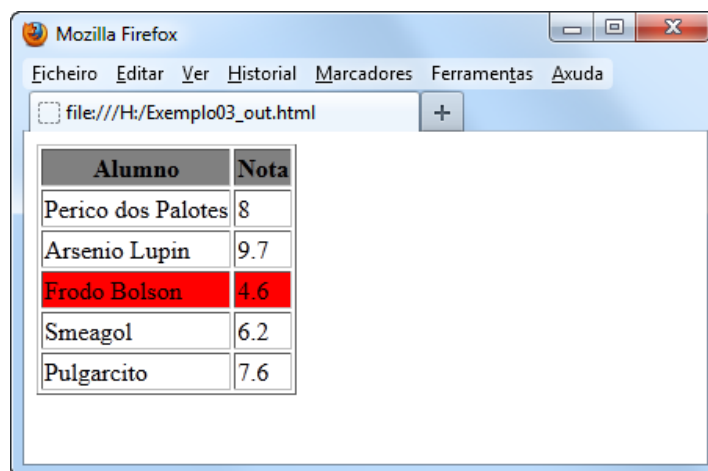
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="UTF-8" indent="yes" method="html"/>
  <xsl:template match="/">
    <html>
      <head></head>
      <body>
```

```

<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 "&lt;" e "&gt;".

### 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 sexa 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>", procesarase 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">

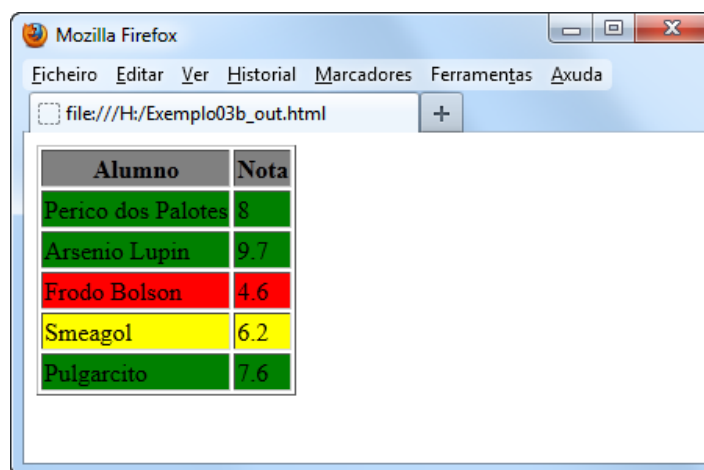
```

```

<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:



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

## 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 ca-



da 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>
  </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:



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.

## Patróns con nome

Ata o de agora traballamos con patróns que o procesador XSLT executaba cando o seu atributo "match" correspondíase cun nodo ou nodos do documento XML orixe. Pero existe outro xeito de executar un patrón: dándolle un nome e dicíndolle ao procesador XSLT que o procese.

Neste caso o patrón deberá ter un atributo "name" que o identifique, e poderá non ter atributo "match" (debe ter un cando menos, pero tamén pode ter ambos atributos). Os patróns que identifiquemos cun atributo "name" poderán executarse dende calquera punto doutro patrón empregando un elemento "<xsl:call-template>" que o identifique co seu propio atributo "name".

Empregar patróns con nome é semellante a empregar funcións nunha linguaxe de programación. Así, no exemplo anterior poderíamos crear un patrón para introducir na táboa os valores de cada fila:

```
<xsl:template name="fila">
  <td><xsl:value-of select="nome" /></td>
  <td><xsl:value-of select="nota" /></td>
</xsl:template>
```

E o executaríamos dende o patrón anterior substituíndo as liñas anteriores por:

```
<xsl:call-template name="fila" />
```

## Parámetros

Os patróns teñen tamén a posibilidade de recibir parámetros cando son executados. Isto é, dentro dun elemento "`<xsl:apply-templates>`" ou dun elemento "`<xsl:call-template>`" podemos indicar o nome e valor dunha ou varias variables que recibirá o patrón que se execute.

Cando creemos un patrón que reciba parámetros, deberemos indicalo empregando un elemento "`<xsl:param>`" por cada un dos parámetros, no que figure o seu nome nun atributo "name".

Cando fagamos a chamada ao patrón, cada un dos parámetros que lle pasemos deberá ir como primeiro contido do elemento "`<xsl:apply-templates>`" ou "`<xsl:call-template>`", no seu propio elemento "`<xsl:with-param>`", que funciona do mesmo xeito que vimos antes coas variables: indicando o seu nome cun atributo "name" e o seu valor ben cun atributo "select" ou ben definíndoo co seu propio contido.

Por exemplo, se queremos que o patrón con nome anterior reciba o nome e a nota do alumno como parámetros, teríamos que definilo do seguinte xeito:

```
<xsl:template name="fila">
  <xsl:param name="nome" />
  <xsl:param name="nota" />
  <td><xsl:value-of select="$nome" /></td>
  <td><xsl:value-of select="$nota" /></td>
</xsl:template>
```

Fíxate que agora os atributos "select" dos elementos "`<xsl:value-of>`" fan referencia a parámetros, e debemos indicalo antepoñéndolles un signo "\$".

Agora na chamada debemos pasarlle ao patrón os valores dos parámetros requiridos.

```
<xsl:call-template name="fila">
  <xsl:with-param name="nome"><xsl:value-of select="nome" /></xsl:with-param>
  <xsl:with-param name="nota"><xsl:value-of select="nota" /></xsl:with-param>
</xsl:call-template>
```

## Valores por defecto

Do mesmo xeito que facemos nunha chamada para definir o valor dun parámetro no elemento "`<xsl:with-param>`", dentro dun patrón podemos definir un valor por defecto de cada parámetro. Por exemplo:

```
<xsl:template name="fila">
  <xsl:param name="nome">(descoñecido)</xsl:param>
  <xsl:param name="nota">(non presentado)</xsl:param>
  <td><xsl:value-of select="$nome" /></td>
  <td><xsl:value-of select="$nota" /></td>
</xsl:template>
```

Estes valores empregaranse cando na chamada ao patrón non se defina o parámetro respectivo.

## Parámetros globais

En XSLT tamén poden crearse parámetros globais, isto é, parámetros que afectan á totalidade do documento XSLT. Para isto, defínese un elemento "`<xsl:param>`" como fillo directo do elemento raíz "`<xsl:stylesheet>`". Estes parámetros poden ter valores por defecto do mesmo xeito que os parámetros dos patróns.

Os parámetros globais poden empregarse para cambiar o comportamento dunha transformación. A forma de indicar os nomes e valores dos parámetros globais cando se executa unha transformación depende do procesador que empreguemos. En Saxon podemos facelo engadindo os nomes dos parámetros e os seus valores a continuación do nome dos documentos XML e XSLT:

```
saxon alumnos.xml taboa_notas.xsl cor_fondo=#e4e4e4
```

En Xalan o procedemento consiste en engadir por cada parámetro unha opción "`-param`" seguida do seu nome e do seu valor. Por exemplo, "`-param cor_fondo #e4e4e4`".

## Eliminando e mantendo espazos

No seu comportamento por defecto, un procesador XSLT mantén os espazos que contén o documento orixe. Se empregamos unha transformación como a seguinte, que simplemente copia un documento orixe directamente ao de saída.

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

O documento resultante será unha copia idéntica do documento orixe, incluíndo aos nodos que conteñan soamente texto.

Para poder eliminar os espazos innecesarios do documento orixe, podemos empregar o elemento XSLT "`<xsl:strip-space>`". Este elemento sitúase como fillo directo do elemento "`<xsl:stylesheet>`" e leva un único atributo "`elements`", no que se indica unha lista separada por espazos de elementos do documento orixe.

O procesador XSLT eliminará os espazos destes elementos antes de procesalos. Tamén podemos incluír na lista de elementos comodíns como o "\*" para indicar varios nodos.

Por exemplo, tomando como documento orixe o seguinte.

```
<?xml version="1.0" encoding="UTF-8"?>
<novas>
  <nova>
    <título>Microsoft multado por non permitir escoller navegador en Windows 7 Service Pack 1</título>
    <contido>As autoridades europeas da Competencia, veñen de multar a <empresa>Microsoft</empresa> por non incluír a posibilidade de escoller navegadores alternativos (a <navegador>IE Explorer</navegador>) en <sistema_operativo>Windows</sistema_operativo> <versión>7</versión> <release>Service Pack 1</release>, aínda que se tiña comprometido a elo.</contido>
  </nova>
</novas>
```

Podemos aplicar 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:strip-space elements="*" />
  <xsl:template match="/">
    <xsl:copy-of select="." />
  </xsl:template>
</xsl:stylesheet>
```

E obteremos a seguinte saída:

```
<?xml version="1.0" encoding="UTF-8"?><novas><nova><título>Microsoft multado por
non permitir escoller navegador en Windows 7 Service Pack 1</título><contido>As
autoridades europeas da Competencia, veñen de multar a <empre-
sa>Microsoft</empresa> por non incluír a posibilidade de escoller navegadores al-
ternativos (a <navegador>IE Explorer</navegador>) en <siste-
ma_operativo>Windows</sistema_operativo><versión>7</versión><release>Service Pack
1</release>, aínda que se tiña comprometido a elo.</contido></nova></novas>
```

En ocasións queremos eliminar tódolos espazos dun documento agás algúns. Por exemplo, o documento orixe anterior contén algúns nodos que conteñen soamente espazos entre "<sistema\_operativo>", "<versión>" e "<release>". Estes nodos forman parte do texto da nova e deberían preservarse.

Neste caso podemos botar man do elemento XSLT "<xsl:preserve-space>", semellante a "<xsl:strip-space>" pero cun significado oposto.

Cando un mesmo elemento está referenciado por un elemento "<xsl:strip-space>" e por outro elemento "<xsl:preserve-space>", aplícase aquela na que o contido do atributo "elements" sexa máis específico.

Por exemplo, cambiando a transformación anterior pola seguinte.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:strip-space elements="*" />
  <xsl:preserve-space elements="contido" />
  <xsl:template match="/">
    <xsl:copy-of select="." />
  </xsl:template>
</xsl:stylesheet>
```

Agora manteranse os espazos dentro do elemento "<contido>".

```
<?xml version="1.0" encoding="UTF-8"?><novas><nova><título>Microsoft multado por
non permitir escoller navegador en Windows 7 Service Pack 1</título><contido>As
autoridades europeas da Competencia, veñen de multar a <empre-
sa>Microsoft</empresa> por non incluír a posibilidade de escoller navegadores al-
ternativos (a <navegador>IE Explorer</navegador>) en <siste-
ma_operativo>Windows</sistema_operativo> <versión>7</versión> <release>Service
Pack 1</release>, aínda que se tiña comprometido a elo.</contido></nova></novas>
```

## Claves e identificadores

Ao procesar un documento XML, podemos crear claves para identificar elementos do documento orixinal. Os procesadores XSLT soen crear índices internos para as claves, o que mellora o acceso aos elementos.

Para crear unha clave emprégase o elemento "<xsl:key>". Este elemento debe ser fillo directo de "<xsl:stylesheet>". Non pode empregarse en ningún outro sitio, como por exemplo formando parte dun patrón.

Ten 3 atributos requiridos:

- "name". Asigna un nome á clave para poder empregala con posterioridade.
- "match". Indica os nodos que serán indexados pola clave.
- "use". Define o elemento ou a propiedade dos nodos que se empregará para crear o índice. O valor dese elemento ou propiedade será o que usaremos para acceder aos nodos do índice.

Por exemplo, se quixéramos crear unha clave para as máquinas do documento XML orixe que figura na tarefa 5, empregando a propiedade "nome", faríamos:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:key name="c_máquinas" match="máquina" use="@nome" />
  ...
```

### Acceder aos nodos da clave

Unha vez creada a clave, empregando a función "key" obtemos nodos da clave. Debemos indicar como parámetros o nome da clave, e o valor do elemento ou propiedade definido no atributo "use".

Por exemplo, se máis adiante (dentro dun patrón) quixéramos obter o "fabricante" da máquina con nome "COPERNICO", poderíamos facer:

```
<xsl:value-of select="key('c_máquinas', 'COPERNICO')/hardware/fabricante" />
```

Cando o valor da clave non é único, a función "key" devolverá o conxunto de nodos que se corresponden co valor que indicamos, non un único nodo. Por exemplo, coa clave:

```
<xsl:key name="t_máquinas" match="máquina" use="tipo" />
```

A función "key('t\_máquinas', 'Semitorre')" devolverá tódolos nodos correspondentes a aquelas máquinas de tipo "Semitorre".

### Agrupamento de nodos

Unha aplicación interesante das claves é o agrupamento de nodos que non están directamente relacionados no documento orixe.

Para facelo necesitamos coñecer unha función específica de XSLT: "generate-id". Esta función recibe como parámetro un nodo, e devolve un valor único xerado aleatoriamente. Este valor será único en todo o documento, e ademais será sempre o mesmo para ese nodo e esa execución, é dicir, se máis adiante empregamos de novo a función "generate-id" co mesmo nodo obteremos o mesmo valor.

Esta función pode empregarse por exemplo para xerar identificadores e enlaces internos aos mesmos nun documento HTML.

```
...
<!-- Poñemos unha áncora -->
<a name="{generate-id(.)}">
  <xsl:value-of select="nome" />
</a>
...
<!-- e xeramos un enlace á ancara -->
<a href="#{generate-id(.)}">Para ir ao elemento, preme aquí.</a>
...
```

Para crear grupos de nodos empregando claves e a función "generate-id", deberemos seguir estes pasos:

- Definir unha clave para a propiedade ou elemento que queremos empregar para agrupar.
- Seleccionar os primeiros nodos de cada un dos grupos que imos facer (os primeiros nodos para cada valor distinto da clave).
- Seleccionar os nodos co mesmo valor que o nodo do paso anterior.

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>7</nota>
  </alumno>
  <alumno>
    <nome>Frodo Bolson</nome>
    <nota>7</nota>
  </alumno>
  <alumno>
    <nome>Smeagol</nome>
    <nota>6</nota>
  </alumno>
  <alumno>
    <nome>Pulgarcito</nome>
    <nota>8</nota>
  </alumno>
</alumnos>
```

Queremos facer un documento de saída con un grupo por cada valor da nota. Por tanto, debemos definir a seguinte clave:

```
<xsl:key name="c_alumno" match="alumno" use="nota"/>
```

Despois seleccionaremos o primeiro nodo de cada valor, isto é, o primeiro nodo co valor 8, o primeiro nodo co valor 7, e o primeiro nodo co valor 6. Para isto necesitamos unha expresión un tanto complexa:

```
<xsl:for-each select="/alumnos/alumno[generate-id(.)=generate-id(key('clave_alumno',nota))]">
```

O funcionamento da expresión é o seguinte:

- O bucle recorre tódolos nodos dos alumnos, quedándose soamente con aqueles nos que se cumpra a condición da expresión XPath.
- A función "key('clave\_alumno',nota)" devolve tódolos alumnos coa mesma nota do nodo que esta a procesar o bucle.
- Cando lle aplicamos a función "generate-id(key('clave\_alumno',nota))" obtemos o ID único correspondente ao primeiro nodo con esa mesma nota.

- Por último, se o ID do nodo que estamos a procesar coincide co ID de ese nodo (é o mesmo nodo), cómprese a condición e se procesa o nodo (é o primeiro nodo correspondente ao grupo de nodos coa súa mesma nota).

Para rematar, dentro deste bucle temos que empregar outro bucle para percorre tódolos nodos coa súa mesma nota. Para isto podemos empregar a función "key":

```
<xsl:for-each select="key('clave_alumno',nota)">
```

A transformación completa, ordenando os grupos de maior a menor nota, podería quedar entón:

```
<?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:key name="clave_alumno" match="alumno" use="nota"/>

  <xsl:template match="/">
    <xsl:element name="Notas">
      <xsl:for-each select="/alumnos/alumno[generate-id(.)=generate-id(key('clave_alumno',nota))]">
        <xsl:sort select="nota" order="descending" />
        <xsl:element name="Nota">
          <xsl:attribute name="valor">
            <xsl:value-of select="nota" />
          </xsl:attribute>
          <xsl:for-each select="key('clave_alumno',nota)">
            <xsl:element name="Alumno">
              <xsl:value-of select="nome" />
            </xsl:element>
          </xsl:for-each>
        </xsl:element>
      </xsl:for-each>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

E ao realizar a transformación, obteríamos:

```
<?xml version="1.0" encoding="UTF-8"?>
<Notas>
  <Nota valor="8">
    <Alumno>Perico dos Palotes</Alumno>
    <Alumno>Pulgarcito</Alumno>
  </Nota>
  <Nota valor="7">
    <Alumno>Arsenio Lupin</Alumno>
    <Alumno>Frodo Bolson</Alumno>
  </Nota>
  <Nota valor="6">
    <Alumno>Smeagol</Alumno>
  </Nota>
</Notas>
```



Para rematar faremos as tarefas 3 e 4, onde veremos algúns patróns avanzados para realizar transformacións XSLT.



## 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.
- Tarefa 3. Transformacións avanzadas. Nesta tarefa traballaremos con transformacións que empreguen algúns aspectos avanzados no procesamento da información do documento XML orixe.
- Tarefa 4. Transformacións avanzadas (II). Nesta tarefa, semellante á anterior, traballaremos de novo con transformacións que empreguen aspectos avanzados no procesamento da información do documento XML orixe.

### 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

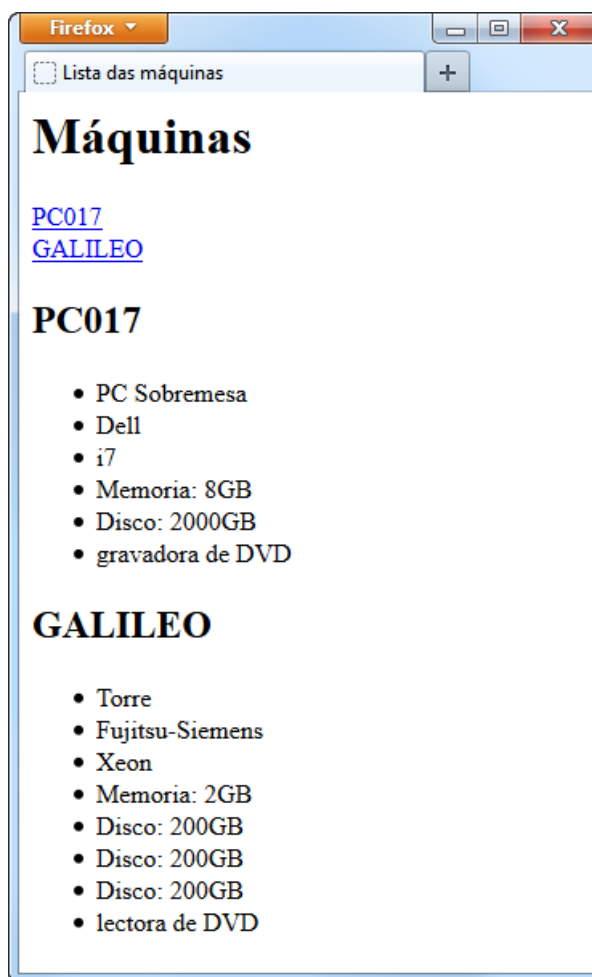
Obter un listado en formato XML dos discos e das memorias que figuran no documento XML orixe como o seguinte:

```
<?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:



## Solución

### 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" />
            <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>
    </máquina>
</equipos>

```

```

<config>
  <OS>Windows XP</OS>
</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 tecnoloxía="DDR">2</memoria>
    <disco tecnoloxía="SCSI" capacidade="500"/>
    <disco tecnoloxía="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>
</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>
<máquina nome="KEPLER">
  <hardware>
    <tipo>Rack</tipo>
    <fabricante>HP</fabricante>
    <procesador marca="Intel" num_nucleos="2" velocidade="3">Core2
Duo</procesador>
    <memoria tecnoloxía="DDR2">4</memoria>
    <disco tecnoloxía="SATA" capacidade="500"/>
    <disco tecnoloxía="SATA" capacidade="500"/>
    <disco tecnoloxía="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>
</máquina>
<máquina nome="NEWTON">
  <hardware>
    <tipo>Rack</tipo>
    <fabricante>HP</fabricante>
    <procesador marca="Intel" num_nucleos="2" velocidade="3">Core2
Duo</procesador>
    <memoria tecnoloxía="DDR2">4</memoria>
    <disco tecnoloxía="SATA" capacidade="500"/>
    <disco tecnoloxía="SATA" capacidade="500"/>
    <gravadora tipo="DVD"/>
  </hardware>

```

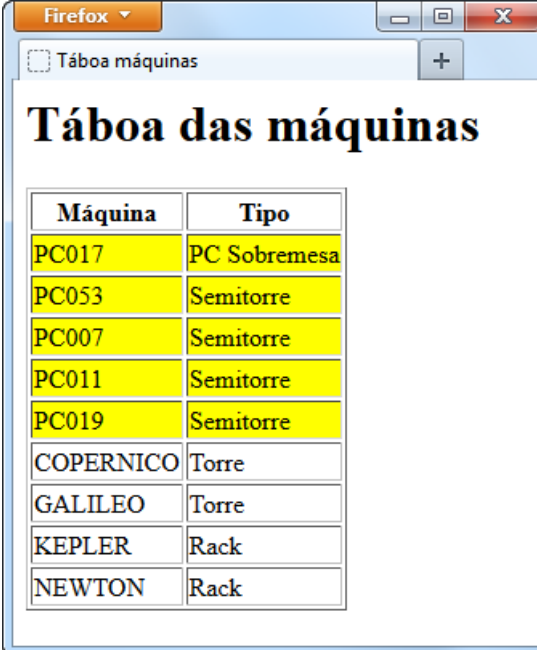
```

<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 Sobre mesa
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 Sobre mesa" 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.

#### d) Tarefa 2\_d

Modificar a transformación anterior empregando:

- Unha variable de nome "cor\_windows" para a cor amarela de fondo das filas das máquinas da familia "Windows".
- Unha variable de nome "encabezado\_taboa" coas etiquetas da fila de encabezado da táboa.
- Unha variable de nome "capacidade" coa capacidade de cada máquina.
- Unha variable de nome "cor\_capacidade" coa cor correspondente ao texto da capacidade de cada máquina.

#### e) Tarefa 2\_e

Transformar o documento orixe noutro documento XML semellante que:

- Inclúa soamente as máquinas cuxo nome comeza por "PC".
- Dentro da lista das máquinas, aparezan en primeiro lugar aquelas cuxo sistema operativo non sexa da familia "Windows".
- As máquinas da familia "Windows" estean ordenadas pola capacidade total de almacenamento, en orde descendente.



## Solución

### 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 Sobremesa'">
        <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, 'Im-
presora'))]">
            <xsl:element name="tr">
              <xsl:if test="hardware/tipo = 'Semitorre' or hardware/tipo = 'PC Sobreme-
sa'">
                <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>
```

```

        </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, 'Impresora'))]">
  <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 solução, 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>

```

#### d) Tarefa 2\_d

Empregando variables, a transformación anterior correspondente ás estruturas iterativas obtida 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:variable name="cor_windows">yellow</xsl:variable>
  <xsl:variable name="encabezado_taboa">
    <tr>
      <th>Máquina</th>
      <th>Tipo</th>
      <th>OS</th>
      <th>Capacidade HD</th>
    </tr>
  </xsl:variable>
  <xsl:template match="/">
    <html>
      <head>
        <title>Táboa de ordenadores con OS</title>
      </head>
      <body>
        <h1>Táboa de ordenadores con OS</h1>
        <table border="1">
          <xsl:copy-of select="$encabezado_taboa" />
          <xsl:for-each select="equipos/máquina[config/OS]" >
            <xsl:sort select="sum(hardware/disco/@capacidade)" order="descending" data-
type="number" />
            <xsl:variable name="capacidade" select="sum(hardware/disco/@capacidade)" />
            <tr>
              <xsl:if test="starts-with(config/OS, 'Windows')">
                <xsl:attribute name="style">background:<xsl:value-of se-
lect="$cor_windows" /></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:variable name="cor_capacidade">
                  <xsl:choose>
                    <xsl:when test="$capacidade >= 1000">red</xsl:when>
                    <xsl:when test="$capacidade >= 500">#FF4500</xsl:when>
                    <xsl:otherwise>orange</xsl:otherwise>
                  </xsl:choose>
                </xsl:variable>
                <xsl:attribute name="style">color:<xsl:value-of select="$cor_capacidade"
/></xsl:attribute>
                <xsl:value-of select="$capacidade" /> GB

```

```

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

```

### e) Tarefa 2\_e

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" method="xml"/>
  <xsl:template match="/">
    <equipos>
      <xsl:apply-templates select="equipos/máquina[starts-with(@nome,
'PC')][not(starts-with(config/OS, 'Windows'))]" />
      <xsl:apply-templates select="equipos/máquina[starts-with(@nome,
'PC')][starts-with(config/OS, 'Windows')]" />
      <xsl:sort select="sum(hardware/disco/@capacidade)" or-
der="descending" data-type="number" />
    </xsl:apply-templates>
  </equipos>
</xsl:template>
<xsl:template match="máquina">
  <xsl:copy-of select="." />
</xsl:template>
</xsl:stylesheet>

```

## 1.3.3 Tarefa 3. Transformacións avanzadas

Tomando como base o mesmo documento XML da tarefa anterior:

### a) Tarefa 3\_a

Queremos obter un documento de saída coa mesma información que o documento XML orixe, pero organizada de forma distinta.

- Baixo o elemento raíz "equipos" haberá un nodo de nome "tipo" por cada tipo de máquina distinto ("Impresora Inxección", "Impresora Láser", "PC Sobremesa", etc.). Cada un destes nodos terá un atributo "nome\_tipo" onde figurará o nome dese tipo.
- Os nodos "tipo" estarán ordenados por orde alfabético segundo o seu nome.
- Dentro de cada nodo "tipo" estarán os nodos "máquina" do documento orixinal que se corresponden con ese tipo concreto. Por exemplo, dentro do tipo "Impresora Inxección" estará a máquina "PR003".
- Cando haxa máis dunha máquina dentro dun tipo, estarán ordenadas polo seu nome.
- O contido de cada máquina será o mesmo que no documento orixe, salvo o elemento "tipo", que xa non é necesario.

Por exemplo, as primeiras liñas do documento de saída que se debe xerar ao aplicar a transformación serán:

```

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

```

```

<tipo nome_tipo="Impresora Inxección">
  <máquina nome="PR003">
    <hardware>
      <fabricante>Lexmark</fabricante>
    </hardware>
    <config/>
  </máquina>
</tipo>
<tipo nome_tipo="Impresora Láser">
  <máquina nome="PR007">
    <hardware>
      <fabricante>OKI</fabricante>
    </hardware>
    <config/>
    <notas>Monocromo, dúplex, rede</notas>
  </máquina>
</tipo>
<tipo nome_tipo="PC Sobremesa">
  <máquina nome="PC017">
    ...

```

### b) Tarefa 3\_b

Agora imos traballar coa configuración de rede das máquinas. Trátase de obter un documento XML no que figuren soamente as máquinas con dirección IP.

Para cada máquina deberánse anotar os datos correspondentes a súa dirección IP e a súa porta de enlace (*gateway*). Ademais haberá que calcular tamén para cada máquina, en función da IP e o *gateway*, a súa máscara de rede, que poderá ser soamente "255.255.255.0" ou "255.255.0.0".

Isto é, teremos que obter un documento XML de saída semellante ao seguinte:

```

<?xml version="1.0" encoding="UTF-8"?>
<equipos>
  <máquina nome="PC017">
    <IP>192.168.10.105</IP>
    <máscara>255.255.0.0</máscara>
    <gateway>192.168.20.1</gateway>
  </máquina>
  <máquina nome="PC011">
    <IP>192.168.10.221</IP>
  </máquina>
  <máquina nome="PC019">
    <IP>192.168.10.45</IP>
    <máscara>255.255.255.0</máscara>
    <gateway>192.168.10.1</gateway>
  </máquina>
  <máquina nome="COPERNICO">
    <IP>192.168.200.11</IP>
    <máscara>255.255.0.0</máscara>
    <gateway>192.168.20.1</gateway>
  </máquina>
  <máquina nome="GALILEO">
    <IP>192.168.20.10</IP>
    <máscara>255.255.255.0</máscara>
    <gateway>192.168.20.1</gateway>
  </máquina>
  <máquina nome="KEPLER">
    <IP>192.168.10.10</IP>
    <máscara>255.255.255.0</máscara>
    <gateway>192.168.10.1</gateway>
  </máquina>

```

```

    <máquina nome="NEWTON">
      <IP>192.168.10.11</IP>
      <máscara>255.255.255.0</máscara>
      <gateway>192.168.10.1</gateway>
    </máquina>
  </equipos>

```

Deberá empregarse un patrón con nome e con parámetros para facer, cando mínimo, o cálculo da máscara de rede correspondente a cada máquina.

### c) Tarefa 3\_c

Para rematar, imos traballar co elemento "notas" que figura nalgúnhas máquinas do documento XML orixe. En algún caso aproveitouse ese elemento para introducir máis dun texto:

```

<notas>Monocromo, dúplex, rede</notas>

```

Imos coller soamente as máquinas que teñen unha nota, e crear unha transformación para que no caso de que teña varios comentarios separados por comas, crear a partir deles unha nota independente por cada uno.

Isto é, trátase de transformar o documento XML orixe no seguinte:

```

<?xml version="1.0" encoding="UTF-8"?>
<equipos>
  <máquina nome="PC007">
    <nota>Sin tarxeta de rede</nota>
  </máquina>
  <máquina nome="PR007">
    <nota>Monocromo</nota>
    <nota>dúplex</nota>
    <nota>rede</nota>
  </máquina>
</equipos>

```

Aconséllase crear un patrón con nome e chamalo de forma recursiva para procesar o elemento "notas" orixinal.

## Solución

### a) Tarefa 3\_a

Unha posible solución é a seguinte:

```

<?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:key name="t_máquinas" match="máquina" use="hardware/tipo" />
  <xsl:template match="/">
    <equipos>
      <!-- Este bucle execútase soamente para a primeira máquina de cada tipo -->
      <xsl:for-each select="equipos/máquina[generate-id(.)=generate-id(key('t_máquinas', hardware/tipo))]">
        <xsl:sort select="hardware/tipo" />
        <xsl:element name="tipo">
          <xsl:attribute name="nome_tipo">
            <xsl:value-of select="hardware/tipo" />
          </xsl:attribute>
          <!-- Co bucle seguinte percorremos tódalas máquinas do mesmo tipo -->

```

```

        <xsl:for-each select="key('t_máquinas',hardware/tipo)">
            <xsl:sort select="@nome" />
            <xsl:element name="máquina">
                <xsl:attribute name="nome">
                    <xsl:value-of select="@nome" />
                </xsl:attribute>
                <xsl:apply-templates />
            </xsl:element>
        </xsl:for-each>
    </xsl:element>
</xsl:for-each>
</equipos>
</xsl:template>
<xsl:template match="hardware">
    <hardware>
        <xsl:apply-templates select="*" />
    </hardware>
</xsl:template>
<xsl:template match="config">
    <config>
        <xsl:apply-templates />
    </config>
</xsl:template>
<xsl:template match="tipo" />
<xsl:template match="*">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:stylesheet>

```

### b) Tarefa 3\_b

Unha posible solución é a seguinte:

```

<?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="/">
        <equipos>
            <xsl:apply-templates select="equipos/máquina[config/IP]" />
        </equipos>
    </xsl:template>
    <xsl:template match="máquina">
        <máquina>
            <xsl:attribute name="nome">
                <xsl:value-of select="@nome" />
            </xsl:attribute>
            <xsl:copy-of select="config/IP" />
            <xsl:call-template name="máscara_red">
                <xsl:with-param name="ip">
                    <xsl:value-of select="config/IP" />
                </xsl:with-param>
                <xsl:with-param name="gateway">
                    <xsl:value-of select="config/gateway" />
                </xsl:with-param>
            </xsl:call-template>
            <xsl:copy-of select="config/gateway" />
        </máquina>
    </xsl:template>
    <xsl:template name="máscara_red">
        <xsl:param name="ip" />
        <xsl:param name="gateway" />
        <xsl:if test="$gateway!=''">

```



```

<!-- Gardamos a lonxitude da dirección IP nunha variable -->
<xsl:variable name="long_ip" select="string-length($ip)" />

<!-- Obtemos a dirección IP ata o último byte -->
<xsl:variable name="ip_parcial">
  <xsl:choose>
    <xsl:when test="contains(substring($ip, $long_ip -1, 2), '.')">
      <xsl:value-of select="substring($ip, 1, $long_ip -1)" />
    </xsl:when>
    <xsl:when test="contains(substring($ip, $long_ip -2, 3), '.')">
      <xsl:value-of select="substring($ip, 1, $long_ip -2)" />
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="substring($ip, 1, $long_ip -3)" />
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>

<!-- Obtemos o valor da máscara: 255.255.255.0 ou 255.255.0.0 -->
<xsl:variable name="máscara">
  <xsl:choose>
    <xsl:when test="contains($gateway, $ip_parcial)">
      <xsl:value-of select="'255.255.255.0'" />
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="'255.255.0.0'" />
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>

  <máscara><xsl:value-of select="$máscara" /></máscara>
</xsl:if>
</xsl:template>
</xsl:stylesheet>

```

### c) Tarefa 3\_c

A solución proposta é a seguinte:

```

<?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="/">
    <equipos>
      <xsl:apply-templates select="equipos/máquina[notas]" />
    </equipos>
  </xsl:template>
  <xsl:template match="máquina">
    <xsl:element name="máquina">
      <xsl:attribute name="nome">
        <xsl:value-of select="@nome" />
      </xsl:attribute>
      <xsl:call-template name="notas">
        <xsl:with-param name="nota">
          <xsl:value-of select="notas"/>
        </xsl:with-param>
      </xsl:call-template>
    </xsl:element>
  </xsl:template>
  <xsl:template name="notas">
    <xsl:param name="nota"/>
    <xsl:if test="contains($nota, ',')">

```

```

        <xsl:element name="nota">
            <xsl:value-of select="substring-before($nota, ',')" />
        </xsl:element>
        <xsl:call-template name="notas">
            <xsl:with-param name="nota">
                <xsl:value-of select="substring-after($nota, ', ')" />
            </xsl:with-param>
        </xsl:call-template>
    </xsl:if>
    <xsl:if test="not(contains($nota, ', '))">
        <xsl:element name="nota">
            <xsl:value-of select="$nota" />
        </xsl:element>
    </xsl:if>
</xsl:template>
</xsl:stylesheet>

```

### 1.3.4 Tarefa 4. Transformacións avanzadas (II)

Tomando como base o seguinte documento XML:

```

<horario ciclo="ASIR Ordinario" grupo="1º" centro="IES San Clemente" ano="2013">
  <horas>
    <hora id="1">
      <inicio>08:45</inicio>
      <fin>09:35</fin>
    </hora>
    <hora id="2">
      <inicio>09:35</inicio>
      <fin>10:25</fin>
    </hora>
    <hora id="3">
      <inicio>10:25</inicio>
      <fin>11:15</fin>
    </hora>
    <hora id="4">
      <inicio>11:15</inicio>
      <fin>12:05</fin>
    </hora>
    <hora id="5">
      <inicio>12:05</inicio>
      <fin>12:55</fin>
    </hora>
    <hora id="6">
      <inicio>12:55</inicio>
      <fin>13:45</fin>
    </hora>
    <hora id="7">
      <inicio>13:45</inicio>
      <fin>14:35</fin>
    </hora>
    <hora id="8">
      <inicio>16:20</inicio>
      <fin>17:10</fin>
    </hora>
    <hora id="9">
      <inicio>17:10</inicio>
      <fin>18:00</fin>
    </hora>
  </horas>
</horario>

```

```

<dia num="1">
  <materia hora="1" nome="Planificación e Administración de Redes" />
  <materia hora="2" nome="Planificación e Administración de Redes" />
  <materia hora="3" nome="Formación e Orientación Laboral" />
  <materia hora="4" nome="Linguaxes de Marcas e Sistemas de Xestión da Informa-
ción" />
  <materia hora="5" nome="Linguaxes de Marcas e Sistemas de Xestión da Informa-
ción" />
  <materia hora="6" nome="Implanación de Sistemas Operativos" />
  <materia hora="7" nome="Implanación de Sistemas Operativos" />
</dia>
<dia num="2">
  <materia hora="1" nome="Linguaxes de Marcas e Sistemas de Xestión da Informa-
ción" />
  <materia hora="2" nome="Linguaxes de Marcas e Sistemas de Xestión da Informa-
ción" />
  <materia hora="3" nome="Xestión de Bases de Datos" />
  <materia hora="4" nome="Xestión de Bases de Datos" />
  <materia hora="5" nome="Xestión de Bases de Datos" />
  <materia hora="6" nome="Implanación de Sistemas Operativos" />
  <materia hora="7" nome="Implanación de Sistemas Operativos" />
</dia>
<dia num="3">
  <materia hora="1" />
  <materia hora="2" nome="Linguaxes de Marcas e Sistemas de Xestión da Informa-
ción" />
  <materia hora="3" nome="Formación e Orientación Laboral" />
  <materia hora="4" nome="Planificación e Administración de Redes" />
  <materia hora="5" nome="Planificación e Administración de Redes" />
  <materia hora="6" nome="Fundamentos Hardware" />
  <materia hora="7" nome="Fundamentos Hardware" />
  <materia hora="8" nome="Implanación de Sistemas Operativos" />
  <materia hora="9" nome="Implanación de Sistemas Operativos" />
</dia>
<dia num="4">
  <materia hora="1" nome="Planificación e Administración de Redes" />
  <materia hora="2" nome="Planificación e Administración de Redes" />
  <materia hora="3" nome="Implanación de Sistemas Operativos" />
  <materia hora="4" nome="Implanación de Sistemas Operativos" />
  <materia hora="5" nome="Formación e Orientación Laboral" />
  <materia hora="6" nome="Xestión de Bases de Datos" />
  <materia hora="7" nome="Xestión de Bases de Datos" />
</dia>
<dia num="5">
  <materia hora="1" nome="Planificación e Administración de Redes" />
  <materia hora="2" nome="Planificación e Administración de Redes" />
  <materia hora="3" nome="Xestión de Bases de Datos" />
  <materia hora="4" nome="Xestión de Bases de Datos" />
  <materia hora="5" nome="Fundamentos Hardware" />
  <materia hora="6" nome="Fundamentos Hardware" />
  <materia hora="7" nome="Formación e Orientación Laboral" />
</dia>
</horario>

```

#### a) Tarefa 4\_a

Queremos obter un documento de saída que soamente conteña a información dos elementos "<dia>", engadíndolle a cada subelemento "<materia>" dous atributos "inicio" e "fin" coas horas nas que comeza e finaliza respectivamente. Isto é, o documento XML de saída debe ser como o seguinte extracto:

```
<?xml version="1.0" encoding="UTF-8"?>
<materias>
  <dia num="1">
    <materia hora="1" nome="Planificación e Administración de Redes"
      inicio="08:45" fin="09:35"/>
    <materia hora="2" nome="Planificación e Administración de Redes"
      inicio="09:35" fin="10:25"/>
    <materia hora="3" nome="Formación e Orientación Laboral" inicio="10:25"
      fin="11:15"/>
    <materia hora="4" nome="Linguaxes de Marcas e Sistemas de Xestión da Informa-
ción"
      inicio="11:15" fin="12:05"/>
    <materia hora="5" nome="Linguaxes de Marcas e Sistemas de Xestión da Informa-
ción"
      inicio="12:05" fin="12:55"/>
    <materia hora="6" nome="Implanación de Sistemas Operativos" inicio="12:55"
      fin="13:45"/>
    <materia hora="7" nome="Implanación de Sistemas Operativos" inicio="13:45"
      fin="14:35"/>
  </dia>
  ...
</materias>
```

#### b) Tarefa 4\_b

Agora, empregando o mesmo documento XML orixe, imos crear como saída unha páxina web que amose o horario completo do curso, co seguinte formato:



	Luns	Martes	Mércores	Xoves	Venres
De 08:45 a 09:35	Planificación e Administración de Redes	Linguaxes de Marcas e Sistemas de Xestión da Información		Planificación e Administración de Redes	Planificación e Administración de Redes
De 09:35 a 10:25	Planificación e Administración de Redes	Linguaxes de Marcas e Sistemas de Xestión da Información	Linguaxes de Marcas e Sistemas de Xestión da Información	Planificación e Administración de Redes	Planificación e Administración de Redes
De 10:25 a 11:15	Formación e Orientación Laboral	Xestión de Bases de Datos	Formación e Orientación Laboral	Implanación de Sistemas Operativos	Xestión de Bases de Datos
De 11:15 a 12:05	Linguaxes de Marcas e Sistemas de Xestión da Información	Xestión de Bases de Datos	Planificación e Administración de Redes	Implanación de Sistemas Operativos	Xestión de Bases de Datos
De 12:05 a 12:55	Linguaxes de Marcas e Sistemas de Xestión da Información	Xestión de Bases de Datos	Planificación e Administración de Redes	Formación e Orientación Laboral	Fundamentos Hardware

#### c) Tarefa 4\_c

E empregando unha vez máis o mesmo documento orixe, imos crear como saída unha lista das materias (por orde alfabético) que figuran nel. Isto é, queremos obter o seguinte:

```
<?xml version="1.0" encoding="UTF-8"?>
<materias>
  <materia>Formación e Orientación Laboral</materia>
  <materia>Fundamentos Hardware</materia>
  <materia>Implanación de Sistemas Operativos</materia>
  <materia>Linguaxes de Marcas e Sistemas de Xestión da Información</materia>
  <materia>Planificación e Administración de Redes</materia>
  <materia>Xestión de Bases de Datos</materia>
</materias>
```

## Solución

### a) Tarefa 4\_a

Unha posible solución é a seguinte:

```
<?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:attribute-set name="atr_materia">
    <xsl:attribute name="hora">
      <xsl:value-of select="@hora" />
    </xsl:attribute>
    <xsl:attribute name="nome">
      <xsl:value-of select="@nome" />
    </xsl:attribute>
  </xsl:attribute-set>
  <xsl:template match="/">
    <materias>
      <xsl:apply-templates select="horario/dia" />
    </materias>
  </xsl:template>
  <xsl:template match="dia">
    <xsl:element name="dia">
      <xsl:attribute name="num">
        <xsl:value-of select="@num" />
      </xsl:attribute>
      <xsl:apply-templates select="materia" />
    </xsl:element>
  </xsl:template>
  <xsl:template match="materia">
    <xsl:element name="materia" use-attribute-sets="atr_materia">
      <xsl:apply-templates select="/horario/horas/hora">
        <xsl:with-param name="id">
          <xsl:value-of select="@hora" />
        </xsl:with-param>
      </xsl:apply-templates>
    </xsl:element>
  </xsl:template>
  <xsl:template match="hora">
    <xsl:param name="id" />
    <xsl:if test="@id=$id">
      <xsl:attribute name="inicio">
        <xsl:value-of select="inicio/text()" />
      </xsl:attribute>
      <xsl:attribute name="fin">
        <xsl:value-of select="fin/text()" />
      </xsl:attribute>
    </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

Outra solución máis sinxela, sen empregar patróns para a hora nin conxuntos de atributos, 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="/">
    <materias>
      <xsl:apply-templates select="horario/dia" />
    </materias>
  </xsl:template>
  <xsl:template match="dia">
    <dia>
      <xsl:attribute name="num">
        <xsl:value-of select="@num" />
      </xsl:attribute>
      <xsl:apply-templates select="materia" />
    </dia>
  </xsl:template>
  <xsl:template match="materia">
    <materia>
      <xsl:variable name="hora">
        <xsl:value-of select="@hora" />
      </xsl:variable>
      <xsl:attribute name="hora">
        <xsl:value-of select="@hora" />
      </xsl:attribute>
      <xsl:attribute name="nome">
        <xsl:value-of select="@nome" />
      </xsl:attribute>
      <xsl:attribute name="inicio">
        <xsl:value-of select="/horario/horas/hora[@id=$hora]/inicio" />
      </xsl:attribute>
      <xsl:attribute name="fin">
        <xsl:value-of select="/horario/horas/hora[@id=$hora]/fin" />
      </xsl:attribute>
    </materia>
  </xsl:template>
</xsl:stylesheet>
```

E aínda máis sinxelo se empregamos a función `current` para evitar a creación da variable 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="xml"/>
  <xsl:template match="/">
    <materias>
      <xsl:apply-templates select="horario/dia" />
    </materias>
  </xsl:template>
  <xsl:template match="dia">
    <dia>
      <xsl:attribute name="num">
        <xsl:value-of select="@num" />
      </xsl:attribute>
      <xsl:apply-templates select="materia" />
    </dia>
  </xsl:template>
  <xsl:template match="materia">
    <materia>
      <xsl:attribute name="hora">
```

```

        <xsl:value-of select="@hora" />
      </xsl:attribute>
      <xsl:attribute name="nome">
        <xsl:value-of select="@nome" />
      </xsl:attribute>
      <xsl:attribute name="inicio">
        <xsl:value-of select="/horario/horas/hora[@id=current()/@hora]/inicio"/>
      </xsl:attribute>
      <xsl:attribute name="fin">
        <xsl:value-of select="/horario/horas/hora[@id=current()/@hora]/fin" />
      </xsl:attribute>
    </materia>
  </xsl:template>
</xsl:stylesheet>

```

#### b) Tarefa 4\_b

Unha posible solución é a seguinte:

```

<?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>
        <title>Horario <xsl:value-of select="horario/@grupo" /> <xsl:value-of se-
lect="horario/@ciclo" /></title>
      </head>
      <body>
        <h1>Horario <xsl:value-of select="horario/@grupo" /> <xsl:value-of se-
lect="horario/@ciclo" />, ano <xsl:value-of select="horario/@ano" /></h1>
        <table border="1">
          <tr>
            <th></th>
            <th>Luns</th>
            <th>Martes</th>
            <th>Mércores</th>
            <th>Xoves</th>
            <th>Venres</th>
          </tr>
          <xsl:apply-templates select="horario/horas/hora" />
        </table>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="hora">
    <tr>
      <th>De <xsl:value-of select="inicio" /><br /> a <xsl:value-of select="fin"
/></th>
      <xsl:apply-templates select="/horario/dia">
        <xsl:with-param name="id"><xsl:value-of select="@id" /></xsl:with-param>
      </xsl:apply-templates>
    </tr>
  </xsl:template>
  <xsl:template match="dia">
    <xsl:param name="id" />
    <td>
      <xsl:value-of select="materia[@hora=$id]/@nome" />
    </td>
  </xsl:template>
</xsl:stylesheet>

```

#### c) Tarefa 4\_c

Unha posible solución é a seguinte:

```
<?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:key name="clave_materia" match="materia" use="@nome"/>
  <xsl:template match="/">
    <materias>
      <xsl:for-each select="/horario/dia/materia[generate-id(.)=generate-
id(key('clave_materia',@nome))]">
        <xsl:sort select="@nome" />
        <xsl:element name="materia">
          <xsl:value-of select="@nome" />
        </xsl:element>
      </xsl:for-each>
    </materias>
  </xsl:template>
</xsl:stylesheet>
```