

1. Transformacións XSLT sinxelas

1.1 Introducción

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

- Identificar a estrutura e o os principais elementos das especificacións de transformación XSLT.
- Crear e empregar especificacións de transformación XSLT sinxelas.

1.2 Actividade

Que é XSL?

CSS = Folla de estilo para HTML

HTML utiliza etiquetas predefinidas.

CSS utilízase para agregar estilos a elementos HTML.

XSL = Folla de estilo para XML

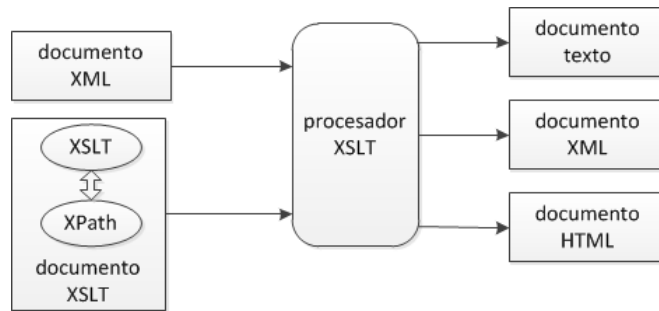
XML non utiliza etiquetas predefinidas e, polo tanto, o significado de cada etiqueta non se comprende ben.

O elemento <table> podería indicar unha táboa HTML, un moble ou outra cousa, je os navegadores non saben cómo mostralo!

Enton, XSL describe cómo deben mostrarse os elementos XML.

O termo **XSL** (*eXtensible Stylesheet Language, Linguaxe de Follas de Estilos Extensible*) xurdiu como agrupación dun conxunto de estándares co obxectivo común de **transformar e aplicar estilos visuais aos documentos XML**. Os estándares que abrangue o termo XSL son:

- **XPath**. Como xa vimos, é unha **linguaxe que serve para construír expresións que busquen e accedan a partes concretas do documento XML**.
- **XSLT** (*XSL Transformations, Transformacións XSL*). É unha linguaxe XML que emprégase xunto con XPath para converter un documento XML nun documento HTML ou XHTML, nun documento de texto, ou noutro documento XML. É dicir, **XSL permite definir a forma en que queremos ver os datos almacenados nun documento XML**. De feito, un documento XML pode ter varias follas de estilo XSL que o mostren en diferentes formatos. En realidade, **XSL é unha linguaxe que define a transformación entre un documento XML de entrada e outro documento XML de saída**.



- **XSL-FO** (*XSL Formatting Objects*, Obxectos de Formato XSL). É **outra linguaxe XML**, pensada neste caso para dar formato aos datos contidos no documento XML. Traballa con áreas, bloques, rexións, anchuras e alturas das páxinas, das marxes, etc. É **moi utilizado para xerar arquivos PDF a partir de documentos XML**.

Nos **imos a estudar as posibilidades da linguaxe XSLT** para realizar transformacións de documentos XML.

Existen tres versións de XSLT:

- XSLT 1.0, que foi publicada polo W3C como recomendación en novembro de 1999.
- XSLT 2.0, que aproveita as vantaxes de XPath 2.0, pero non é moi empregada (recomendación de xaneiro de 2007).
- XSLT 3.0, que a comezos de 2013 segue en desenvolvemento polo W3C (borrador de traballo).

Para facer a transformación precisamos un procesador XSLT, isto é, unha aplicación que comprenda as linguaxes XSLT e XPath e realice sobre o documento XML o procesamento que se indica no documento XSLT.

No proceso de transformación, XSLT usa XPath para definir partes do documento fonte que deben coincidir cunha ou máis plantillas predefinidas. Cando encontra unha coincidencia, XSLT transformará a parte coincidente do documento fonte no documento resultante.

Todoos principais navegadores son compatibles con XSLT e XPath.

Existen outros procesadores XSLT dispoñibles como aplicacións independentes (non integrados nun navegador web).

Transformacións XSLT

Imos comezar cunha transformación sinxela, **tomando como orixe o seguinte documento XML (Exemplo01.xml)**:

```
<?xml version="1.0" encoding="utf-8"?>
<ciclo>
  <módulo sesións="5" horas="133">Linguaxes de marcas</módulo>
  <profesor>Xaime Louzán</profesor>
</ciclo>
```

Empregando un navegador web, **queremos obter unha páxina HTML como a seguinte**:



A especificación da transformación a realizar almacénase nun documento XSLT (normalmente con extensión ".xsl" (*Exemplo01.xsl*).

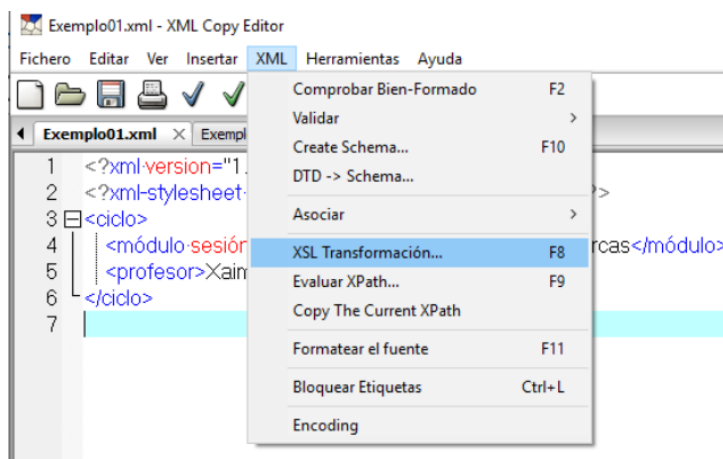
Deberemos vincular o documento XML con este documento XSLT onde se especifica a transformación. Isto faise engadíndolle unha instrución de procesamento como a seguinte:

```
<?xml-stylesheet type="text/xsl" href="Exemplo01.xsl"?>
```

Esta instrución debe figurar dentro do prólogo, concretamente despois da declaración XML e antes do elemento raíz do documento XML. Por tanto, o documento XML orixinal quedaría:

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="Exemplo01.xsl" ?>
<ciclo>
    <módulo sesións="5" horas="133">Linguaxes de marcas</módulo>
    <profesor>Xaime Louzán</profesor>
</ciclo>
```

No caso de empregar outra ferramenta distinta a un navegador para realizar a transformación, tamén é común que exista algún outro xeito de indicar o documento XSLT que queremos aplicar. Por exemplo, o **Editor XML** permite asociar unha folla de estilos XSLT cun documento XML, realizar a transformación e ver o resultado. O resultado da transformación XSLT móstrase nunha nova fiestra de documento.

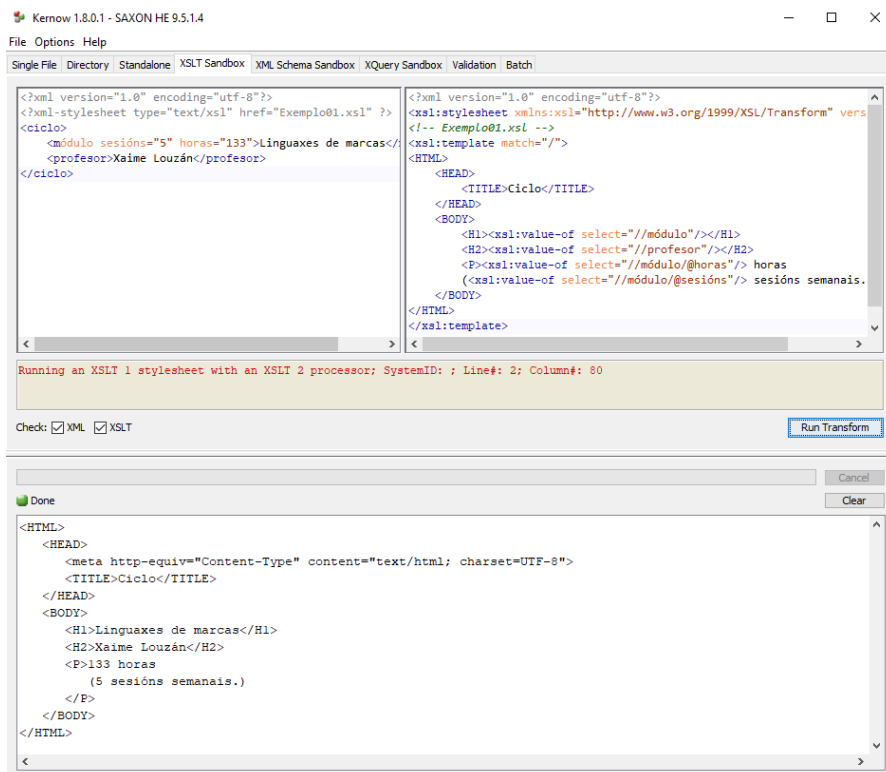


```
<?xml version="1.0" encoding="UTF-8"?>
<HTML>
<HEAD>
  <TITLE>Ciclo</TITLE>
</HEAD>
<BODY>
  <H1>Linguaxes de marcas</H1>
  <H2>Xaime Louzán</H2>
  <P>133 horas
    (5 sesións semanais.)</P>
</BODY>
</HTML>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<HTML>
<HEAD>
  <TITLE>Ciclo</TITLE>
</HEAD>
<BODY>
  <H1>Linguaxes de marcas</H1>
  <H2>Xaime Louzán</H2>
  <P>133 horas
    (5 sesións semanais.)</P>
</BODY>
</HTML>
```



Se empregamos a **ferramenta Kernow** (baseada no procesador XSLT Saxon), podemos **escribir directamente os documentos XML e XSLT e executar a transformación.**



Nalgúns casos queremos empregar un mesmo documento XML orixe para obter varios resultados de saída aplicando distintas transformacións. Por exemplo, a partir dun documento XML cos datos dunha venda, poderíamos xerar unha factura en formato HTML para presentar nun navegador, un documento XML cun resumo da mesma para engadir na contabilidade, e incluso un novo documento HTML específico para visualizar nun terminal móbil.

Tamén deberemos ter en conta que non calquera navegador vai ser capaz de realizar calquera transformación. Por exemplo, o navegador *Chrome* non soporta, por motivos de seguridade, transformacións XSLT cando estean aplicadas a documentos XML locais.

Documentos XSLT

Un documento XSLT **é un documento XML cunha estrutura específica**. Polo tanto, **deberá comezar cun prólogo XML** como o seguinte:

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

O **elemento raíz dun documento XSLT** é normalmente "**<stylesheet>**" (aínda que tamén se pode empregar "**<transform>**", son sinónimos e pódese empregar calquera deles indistintamente), indicando de xeito obrigatorio a versión empregada.

Tamén é preciso facer referencia no documento ao espazo de nomes "**http://www.w3.org/1999/XSL/Transform**". As etiquetas pertencentes a este espazo de nomes serán as que conterán as instrucións destinadas ao procesador XSLT que indican a transformación a realizar.

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

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

E como os documentos XSLT soen conter tamén outras etiquetas non destinadas directamente ao procesador XSLT, **é aconsellable empregar un prefixo para o espazo de nomes** (e non poñelo como espazo de nomes por defecto). Pódese coller calquera prefixo, pero é común empregar "**xsl**" (así figura nos exemplos de aquí en diante), de xeito que **a estrutura base dun documento XSLT** 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:stylesheet>
```

Por exemplo, o documento XSLT da transformación anterior é:

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

<!-- Exemplo01.xsl -->
<xsl:template match="/">
<HTML>
  <HEAD>
    <TITLE>Ciclo</TITLE>
  </HEAD>
  <BODY>
    <H1><xsl:value-of select="//módulo"/></H1>
    <H2>- <xsl:value-of select="//profesor"/> -</H2>
    <P><xsl:value-of select="//módulo/@horas"/> horas
      (<xsl:value-of select="//módulo/@sesións"/> sesións semanais.)</P>
  </BODY>
</HTML>
</xsl:template>
</xsl:stylesheet>
```

Como vemos, *o espazo de nomes "http://www.w3.org/1999/XSL/Transform" emprégase para identificar as instrucións destinadas ao procesador XSLT, de forma que non tente interpretar as outras etiquetas do documento como "<HTML>"*.

Formatos de saída

Coa etiqueta "**<xsl:output>**" definimos as características do documento de saída.

```
<xsl:output
method="xml|html|text"
indent="yes|no"
version="string"
encoding="string"
omit-xml-declaration="yes|no"
standalone="yes|no"
doctype-public="string"
doctype-system="string"
cdata-section-elements="namelist"
media-type="string"
/>
```

Tódolos atributos da etiqueta son opcionais. É importante o atributo "**method**", que **indica o formato do documento resultante**. Se non se inclúe, o formato de saída por defecto é XML (a non ser que o elemento raíz do documento resultante sexa "HTML"; nese caso o formato de saída será HTML se non se indica ningún outro, tal e como sucedía no exemplo anterior).

Cando o documento obtido está en formato HTML, normalmente o procesador XSLT emprega sangrado para mellorar a súa lexibilidade. Isto pódese evitar (ou aplicar tamén a outros formatos de saída) empregando o atributo "*indent*".

Aínda que depende do procesador XSLT que empreguemos, na maioría das ocasións se o documento resultante é XML incluírase no mesmo, de forma automática, unha declaración XML. Este comportamento pódese controlar empregando o atributo "*omit-xml-declaration*".

Os atributos "*version*" e "*encoding*" permiten indicar a versión XML e a codificación a

empregar no documento XML resultante. Os tipos de codificación admitidos dependen do procesador XSLT, aínda que deberán soportar cando menos "utf-8" e "utf-16".

Por exemplo:

Atlas.xml

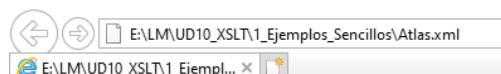
```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="Atlas.xsl" ?>
<Atlas xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespaceSchemaLocation="Atlas1.xsd">
  <Pais>
    <Nombre>Colombia</Nombre>
    <Capital>Bogotá</Capital>
    <Población>450000000</Población>
    <Extensión>1141748</Extensión>
  </Pais>
  <Pais>
    <Nombre>España</Nombre>
    <Capital>Madrid</Capital>
    <Población>416000000</Población>
    <Extensión>505957</Extensión>
  </Pais>
</Atlas>
```

Atlas.xslt

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

Abre o arquivo Atlas.xml cun navegador (Internet Explorer ou Firefox). A maneira mais sinxela é arrastrando o arquivo XML a fiestra do navegador ou abrilo con *Abrir con*.

O resultado debería ser o seguinte:



Colombia
Bogotá
450000000
1141748

España
Madrid
416000000
505957

A liña de código `<xsl:value-of select="Atlas"/>` imprime a base de datos enteira en formato de texto sinxelo. Se examinas os compoñentes da liña, verás por que:

xsl: value- of (*literalmente, valor-de*): é unha instrución que serve para imprimir o valor dun elemento; é dicir, o texto contido entre a etiqueta de inicio e de peche.

select="Atlas" (*en español, selecciona="Atlas"*): esta instrución indica o elemento que contén o valor que debería imprimirse. A menos que declares o contrario, se apuntas cara a un elemento pai (*parent*) o procesador tamén imprimirá o valor dos elementos contidos (*children*). Polo tanto, ao apuntar a raíz obtemos o valor de Nombre, Capital, etc.

Elaboración de transformacions XSLT

- Baséase en considerar que os elementos que forman o documento XML manteñen unha relación xerárquica (*son nodos dunha árbore*).
- Unha folla de estilo XSL consta dunha serie de **regras** que determinan como se debe realizar a transformación. Cada regra componse de:
 - **patrón** (*pattern*).
 - **plantilla** (*template*) ou **acción**.
- Cada regra afecta a un ou varios elementos do documento XML.
- Sintácticamente, as regras teñen tres partes:
 - A marca de apertura que contén un atributo *match* que describe a que partes do documento aplícase a regra (que nodos están afectados). A sintaxe do patrón (valor do atributo *match*) debe seguir as especificacións da linguaxe **XPath**.
 - A parte central describe que debe facerse cando se produce unha coincidencia.
 - A marca de peche.

```
<xsl:template match="PATRÓN">           <!-- Apertura -->
...                                     <!-- Contenido -->
</xsl:template>                         <!-- Cierre -->
```

Así, cada elemento template asociase cun fragmento do documento XML (que pode ser un elemento ou un conxunto de elementos) e se transforma noutro fragmento de XML ou HTML, de acordo ao que especifique no seu interior.

Patróns XSLT

O documento XSLT do exemplo anterior contén o elemento "**<xsl:template>**". Este elementos, patrón, como vimos, é unha parte fundamental das transformacións XSLT.

Os patróns **indican unha transformación a realizar a certos elementos do documento orixinal**. A forma máis habitual de indicar os elementos do documento orixinal aos que se vai a aplicar o patrón, é empregando unha expresión XPath co atributo "*match*".

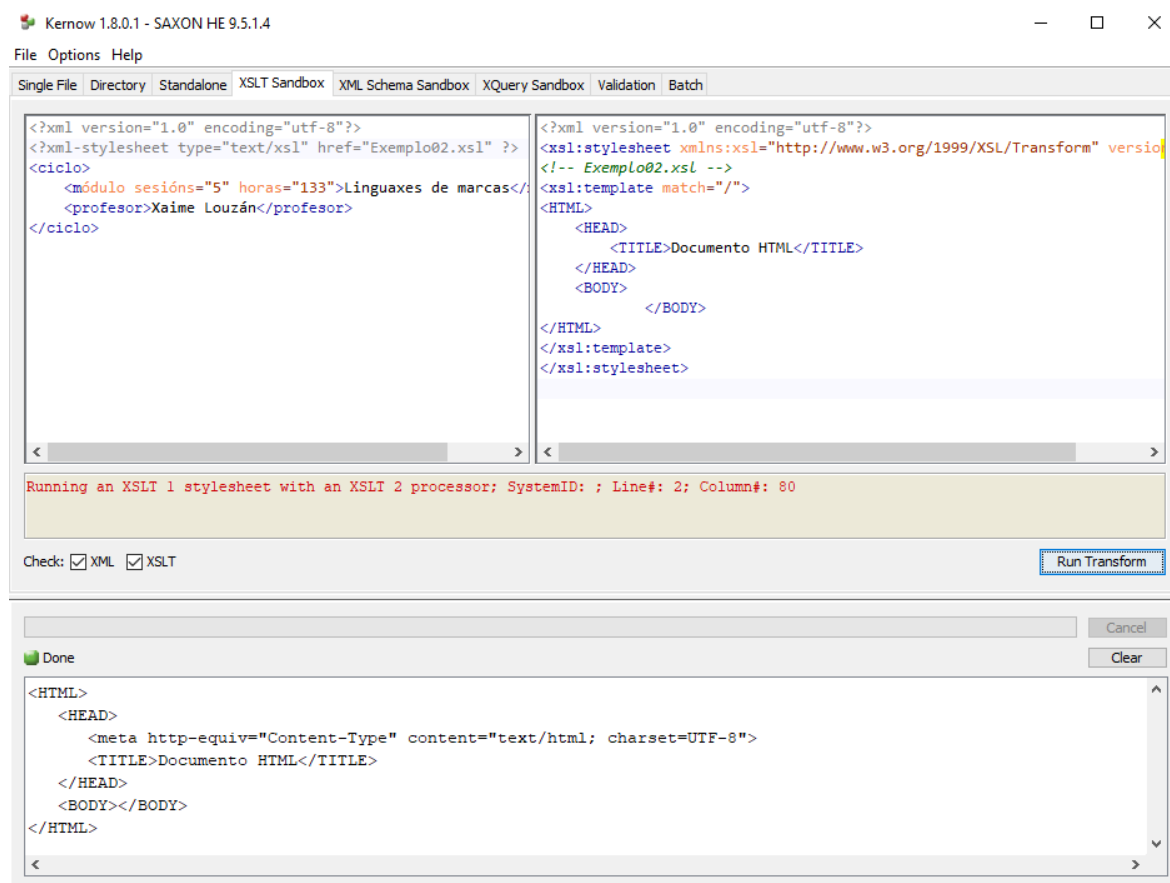
Por **exemplo**, se queremos **aplicar o patrón ao documento XML completo**, faremos tal como fixemos no exemplo anterior:

```
<xsl:template match="/">
...
</xsl:template>
```

O patrón pode conter texto e etiquetas (por exemplo, código en formato HTML), **que pasarán a copiarse no documento de saída**.

Por **exemplo**, o seguinte documento XSLT (*Exemplo02.xsl*) obtén como resultado sempre o mesmo documento HTML baleiro a partires dun documento XML calquera.

```
<?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>
        <TITLE>Documento HTML</TITLE>
      </HEAD>
      <BODY>
      </BODY>
    </HTML>
  </xsl:template>
</xsl:stylesheet>
```



Resumindo, ¿Cómo se realiza a transformación?

- O documento orixe pásase ao procesador XSLT.
 - (O procesador carga unha folla de estilos XSLT)
- O procesador entón:
 - Carga os patróns especificados na folla de estilo ...
 - Recorre o documento XML orixe, nodo por nodo ...
 - Para cada nodo busca un patrón que o referencie no seu atributo "*match*".
 - Unha vez atopado o patrón, aplica a transformación definida no mesmo ao nodo do documento orixe.
 - Proporciona o resultado nun novo documento

Por **exemplo**, obteríamos o mesmo resultado da transformación anterior se tivéramos posto:

```
<xsl:template match="/ciclo">
...
</xsl:stylesheet>
```

É moi **importante ter en conta** que cando o procesador atopa un patrón que fai referencia ao nodo que está a procesar, logo de aplicar a transformación correspondente marca ao nodo e a tódolos seus fillos como procesados, polo que non buscará outro patrón co que transformalos.

Por **exemplo**, se o documento XML do exemplo anterior o transformáramos cun 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="ciclo">
...
</xsl:template>
<xsl:template match="módulo">
...
</xsl:template>
</xsl:stylesheet>
```

O primeiro nodo do documento orixe que procesa é "<ciclo>", e unha vez aplicada a transformación que indica o seu patrón correspondente, os nodos "<módulo>" e "<profesor>" tamén quedan marcados como procesados, polo que o segundo patrón do documento XSLT nunca se executará (xa veremos posteriormente como se levan a cabo as transformacións XSLT con varios patróns).

Patróns predefinidos

Fan que se apliquen uns patróns fixos a aqueles elementos non procesados por ningún outro patrón.

- ➔ Cando empregábamos as expresións "/" ou "/ciclo", estas abranguían a tódolos seus fillos e polo tanto ao documento orixinal completo. Non quedaba ningún ele-

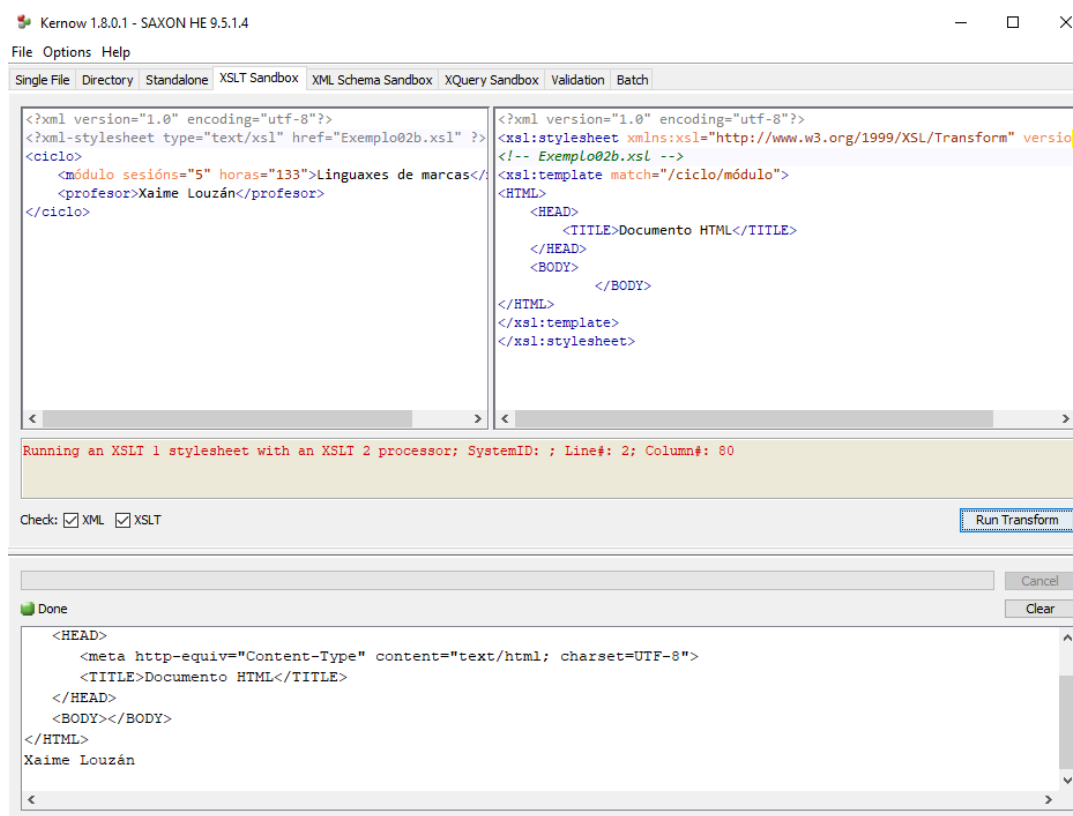
mento do documento orixinal sen procesar.

```
<?xml version="1.0" encoding="UTF-8"?>
<HTML>
<HEAD>
<TITLE>Documento HTML</TITLE>
</HEAD>
<BODY>
</BODY>
</HTML>
```

→ Ao empregar a expresión `"/ciclo/módulo"` no patrón (`<xsl:template match="/ciclo/módulo">`), o resultado obtido non é o que en principio poderíamos esperar. Os elementos `<ciclo>` e `<profesor>` non se atopan cubertos por el; por isto o procesador XSLT aplícalles un patrón predefinido. Basicamente, o comportamento destes patróns predefinidos é procesar o elemento e tódolos seus fillos, copiando ao documento de saída o texto que conteñen.

No noso **exemplo** (*Exemplo02b.xml* y *Exemplo02b.xsl*):

- Ao procesar o elemento `<ciclo>`, o procesador XSLT non atopa un patrón e aplica o patrón predefinido. Este copia ao documento de saída o salto de liña que existe antes do elemento `<módulo>`. Por iso aparece unha liña baleira ao comezo do documento resultante.
- O elemento `<módulo>` ven recollido no patrón `"/ciclo/módulo"`, e ao procesalo cópanse no documento de saída as etiquetas correspondentes ao documento HTML baleiro.
- Por último, o procesador XSLT chega ao elemento `<profesor>` que tampouco ten un patrón específico, co cal copia tamén o seu contido (o nome do profesor) ao documento de saída.



Fíxate que ao contrario que sucede cos patróns que nos definimos no documento XSLT, ao empregar un patrón predefinido non se marcan como procesados os fillos do nodo correspondente do documento XML.



Tarefa 1 (Resolta). Busca dun nodo concreto

Trátase de crear unha transformación XSLT que busque a existencia dun nodo específico no documento XML orixe. En concreto, tomando como orixe o seguinte documento XML:

```
<?xml version="1.0" encoding="utf-8"?>
<venda>
  <cliente cod="CL09384"/>
  <produtos>
    <produto cod="LACT093"/>
    <produto cod="LACT012"/>
    <produto cod="ACEI015"/>
    <produto cod="AUGA005"/>
    <produto cod="CONS121"/>
  </produtos>
</venda>
```

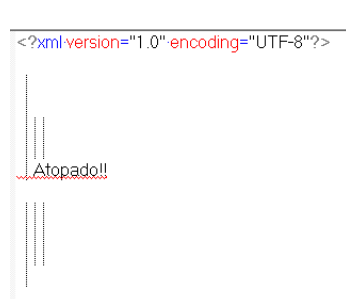
Imos crear un documento XSLT que ao procesalo obteña como **resultado un documento de texto**. Cando no documento orixe existe o produto co atributo "cod="LACT012"", o documento de saída deberá conter o texto "Atopado!!".

Podedes intentar facelo vós, pero por se non vos sae, os deixo a solución

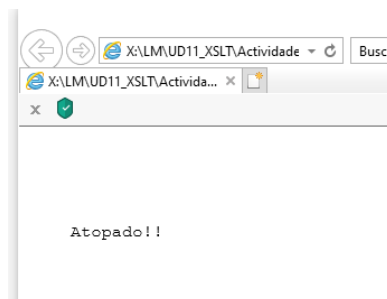
A solución pasa por crear un patrón que se cumpra soamente cando exista o elemento, e nese caso escriba o texto necesario no documento de saída.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text"/>
  <xsl:template match="produto[@cod='LACT012']">
    Atopado!!
  </xsl:template>
</xsl:stylesheet>
```

Salida no XML Copy Editor



Salida no navegador Internet Explorer



Creación de texto e elementos

Existen algúns elementos XSLT que podemos engadir dentro dos patróns, e que nos permiten crear novo contido no documento resultante.

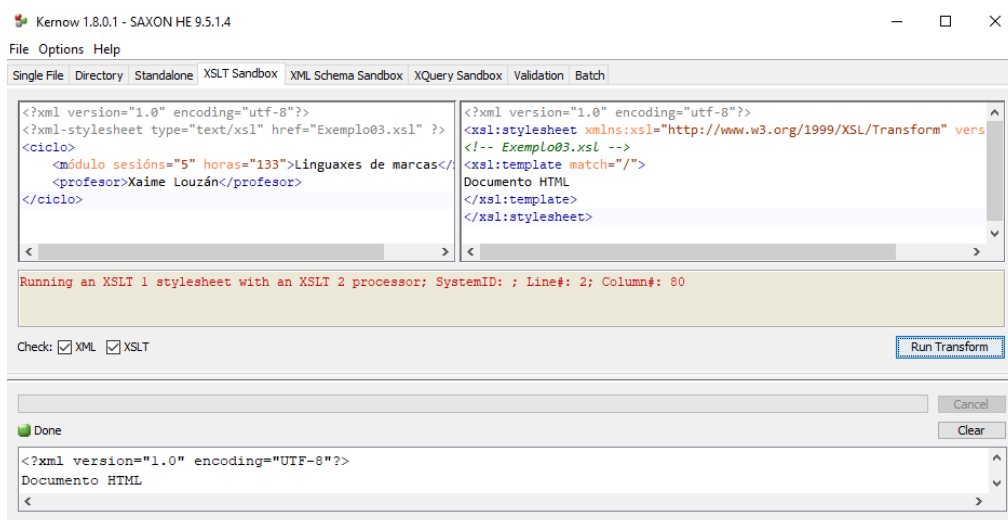
Texto

O elemento "`<xsl:text>`" emprégase para incluír texto no documento resultante. O resultado é semellante a escribir directamente o texto dentro do patrón, pero neste caso temos maior control sobre os espazos e os saltos de liña.

Por exemplo, o resultado da 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="/">
  Documento HTML
</xsl:template>
</xsl:stylesheet>
```

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

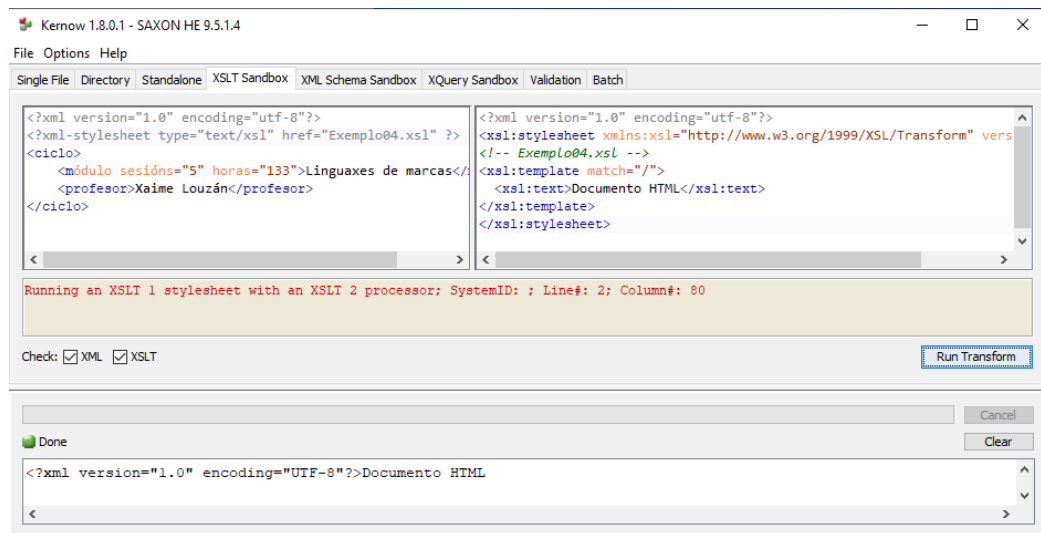


(Exemplo03.xml y Exemplo03.xsl)

e o resultado da seguinte diferéncianse soamente nun salto de liña e uns espazos:

```
<?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:text>Documento HTML</xsl:text>
</xsl:template>
</xsl:stylesheet>
```

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



(Exemplo04.xml y Exemplo04.xsl)

O elemento "`<xsl:text>`" **deberá conter soamente texto, non outras etiquetas.**

Por **exemplo**, o seguinte documento XSLT dará error ao procesalo.

```
<?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:text><HTML></xsl:text>
</xsl:template>
</xsl:stylesheet>
```

Elementos

Dentro dun patrón podemos empregar "`<xsl:element>`" para **crear un novo elemento no documento de saída.** É obrigatorio indicar o nome do novo elemento mediante o atributo "`name`".

Por **exemplo**, se quixeramos obter un documento de saída cun elemento raíz "`<documento>`" baleiro, poderíamos facer:

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

Ou tamén:

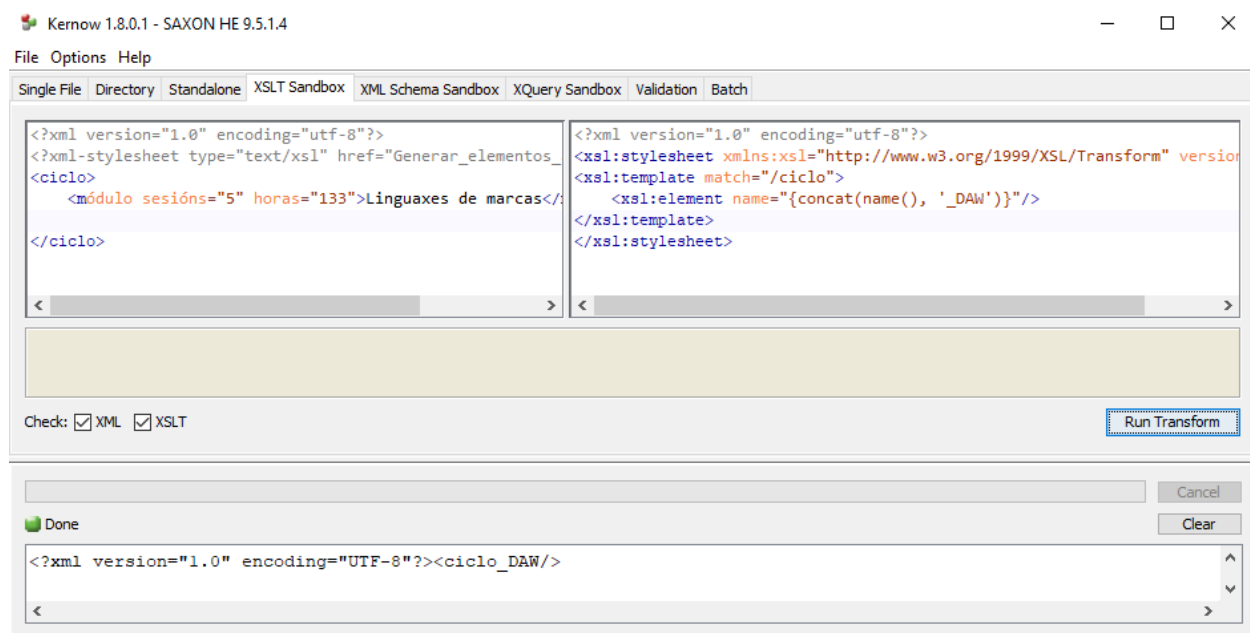
```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <xsl:element name="documento" />
</xsl:template>
</xsl:stylesheet>
```

Un bo motivo para empregar "<xsl:element>" é que o contido do atributo "name" pode estar calculado empregando expresións XPath (por exemplo, a partires de texto, variables, valores retornados por funcións, etc). Neste caso, a expresión debe figurar entre chaves.

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="Generar_elementos.xsl" ?>
<ciclo>
  <módulo sesións="5" horas="133">Linguaxes de marcas</módulo>
</ciclo>
```

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/ciclo">
  <xsl:element name="{concat(name(), '_DAW')}" />
</xsl:template>
</xsl:stylesheet>
```

Obtendo o seguinte resultado:



Atributos

Cando creamos un novo elemento no documento de saída empregando "<xsl:element>", tamén podemos especificar os seus atributos. Simplemente teremos que engadir como fillos deste tantos elementos "<xsl:attribute>" como necesitemos, indicando en cada un deles o seu nome co atributo "name". O seu valor será o texto que conteña.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/ciclo">
  <xsl:element name="{concat(name(), '_DAW')}">
```

```

    <xsl:attribute name="duración">2000 horas</xsl:attribute>
  </xsl:element>
</xsl:template>
</xsl:stylesheet>

```

Obtendo o seguinte resultado:

```

<?xml version="1.0" encoding="UTF-8"?>
<ciclo_DAW duración="2000 horas"/>

```

Selección de valores do documento orixe

Nalgúns casos necesitamos **obter o valor dun elemento ou atributo do documento orixe para empregalo como parte do documento de saída**. Para isto empregaremos "**<xsl:value-of>**", indicando co atributo "select" a expresión XPath que identifique o contido a extraer.

```
<xsl:value-of select="PATRÓN" />
```

Por **exemplo**, se partindo do seguinte documento XML (*SeleccionValores.xml*):

```

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

```

Queremos crear unha transformación que obteña como resultado:

```

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

<profesor nome="Xaime Louzán"/>

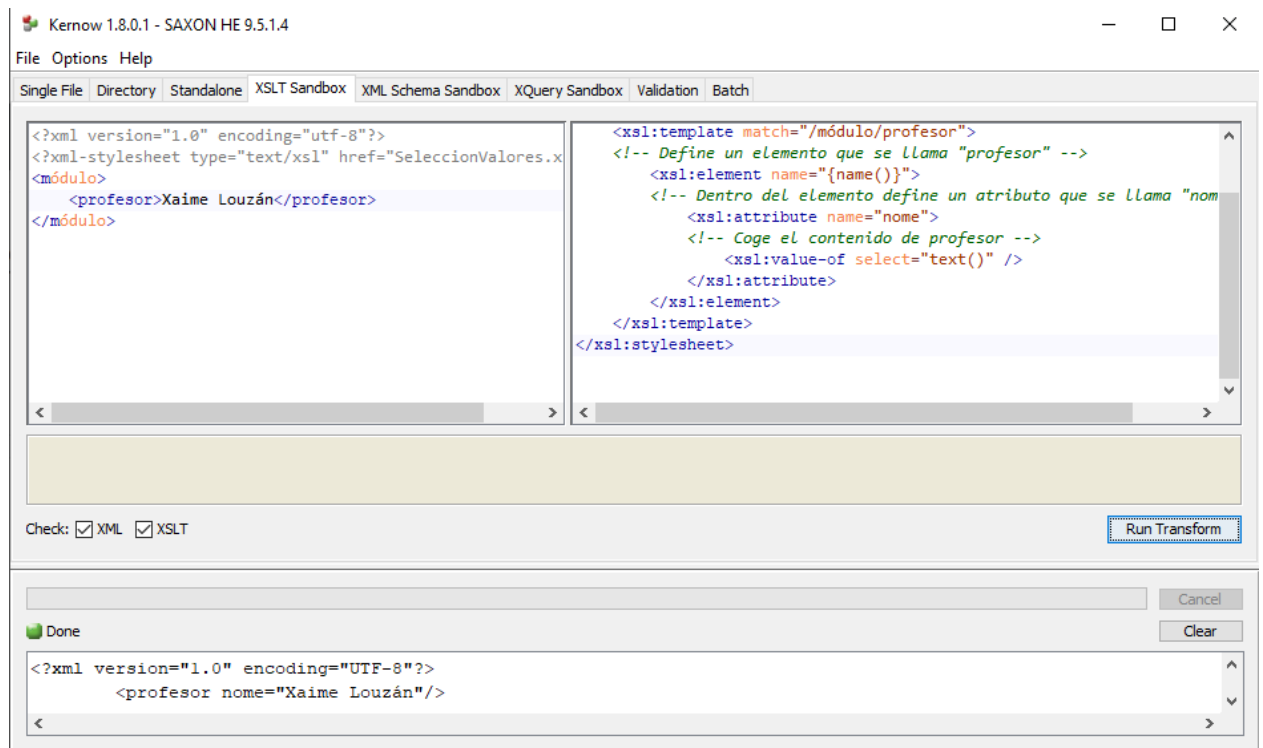
```

Teremos que aplicarlle un documento XSLT como o seguinte (*SeleccionValores.xsl*):

```

<?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">
    <!-- Define un elemento que se llama "profesor" -->
    <xsl:element name="{name()}">
      <!-- Dentro del elemento define un atributo que se llama "nome" -->
      <xsl:attribute name="nome">
        <!-- Coge el contenido de profesor -->
        <xsl:value-of select="text()" />
      </xsl:attribute>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>

```

O elemento "`<xsl:value-of>`" tamén admite o atributo opcional "`disable-output-escaping`", que pode tomar os valores "`yes`" ou "`no`".

Co valor "`yes`" indícase que os caracteres especiais do documento orixe, como "`<`", deben ser pasados tal cual ao documento de saída. Co valor "`no`", que é a opción por defecto, indícase que os caracteres especiais deben ser transformados nas súas respectivas entidades (por exemplo, "`<`").



Tarefa 2. Creación de novos elementos e atributos

Partindo do mesmo documento XML da tarefa1, imos crear algunhas transformacións que obteñan documentos de saída de diversos tipos.

a) Tarefa 2_a

Nesta tarefa obteremos como saída un novo documento XML. Este novo documento XML deberá conter soamente un nodo raíz no que o nome obterase a partires do atributo "`cod`" do cliente. Tamén teremos que crear un atributo para ese elemento con nome "`num_produtos`" que conteña como valor o número de produtos que figuren no documento orixe.

Por exemplo, a transformación deberá transformar o documento anterior no seguinte:

```
<?xml version="1.0" encoding="utf-8"?>
<CL09384 num_produtos="5"/>
```

Intentade facelo vós, pero por se non vos sae, os deixo a solución.

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml"/>
<xsl:template match="venda">
  <xsl:element name="{cliente/@cod}">
    <xsl:attribute name="num_produtos">
      <xsl:value-of select="count(produtos/produto)" />
    </xsl:attribute>
  </xsl:element>
</xsl:template>
</xsl:stylesheet>

```

b) Tarefa 2_b

Agora trátase de obter como saída un documento de texto que conteña unha lista cos códigos dos produtos do documento orixe. Por exemplo:

```

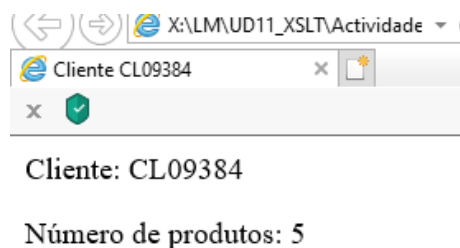
LACT093
LACT012
ACEI015
AUGA005
CONS121

```

De momento é normal que no documento de saída aparezan espazos ou saltos de liña non desexados.

c) Tarefa 2_c

Crear unha especificación de transformación XSLT para obter, a partir do documento XML dado, unha páxina web como a seguinte:



Podedes facer a tarefa de dúas formas:

- Empregando etiquetas HTML embebidas dentro do documento XSLT (creo que é máis fácil).
- Empregando elementos "<xsl:element>", "<xsl:attribute>" e "<xsl:text>" para xerar o documento HTML de saída.

d) Tarefa 2_d

Crear unha transformación que obteña un documento de texto como saída, que transforme cada produto do documento orixe nunha liña co seguinte formato:

```
Produto da familia FAML con código CCC para o cliente de código CL0000
```

Isto é, deberemos obter como documento de saída (ademais de liñas en branco e outros espazos):

```
Produto da familia LACT con código 093 para o cliente de código CL09384
Produto da familia LACT con código 012 para o cliente de código CL09384
Produto da familia ACEI con código 015 para o cliente de código CL09384
Produto da familia AUGA con código 005 para o cliente de código CL09384
Produto da familia CONS con código 121 para o cliente de código CL09384
```

Conxuntos de atributos

Tamén é posible **definir un conxunto de atributos para despois empregalo en un ou varios elementos**. O conxunto de atributos defínese empregando "**<xsl:attribute-set>**". **Débase indicar o nome do conxunto cun atributo "name"**.

Por exemplo:

```
<xsl:attribute-set name="atr_módulo">
  <xsl:attribute name="nome">
    <xsl:value-of select="ciclo/módulo" />
  </xsl:attribute>
  <xsl:attribute name="sesións_anuais">
    <xsl:value-of select="ciclo/módulo/@horas * 1.2"/>
  </xsl:attribute>
</xsl:attribute-set>
```

¡¡OLLO!! A definición do conxunto de atributos debe facerse como fillo directo do elemento raíz "**<xsl:stylesheet>**", fora de calquera patrón.

Para empregar o conxunto na definición dun elemento, engádeselle o atributo "**use-attribute-sets**".

Por exemplo:

```
<xsl:element name="módulo" use-attribute-sets="atr_módulo">
```

Deste xeito, aplicando a seguinte transformación ao documento XML co que vimos traballando.

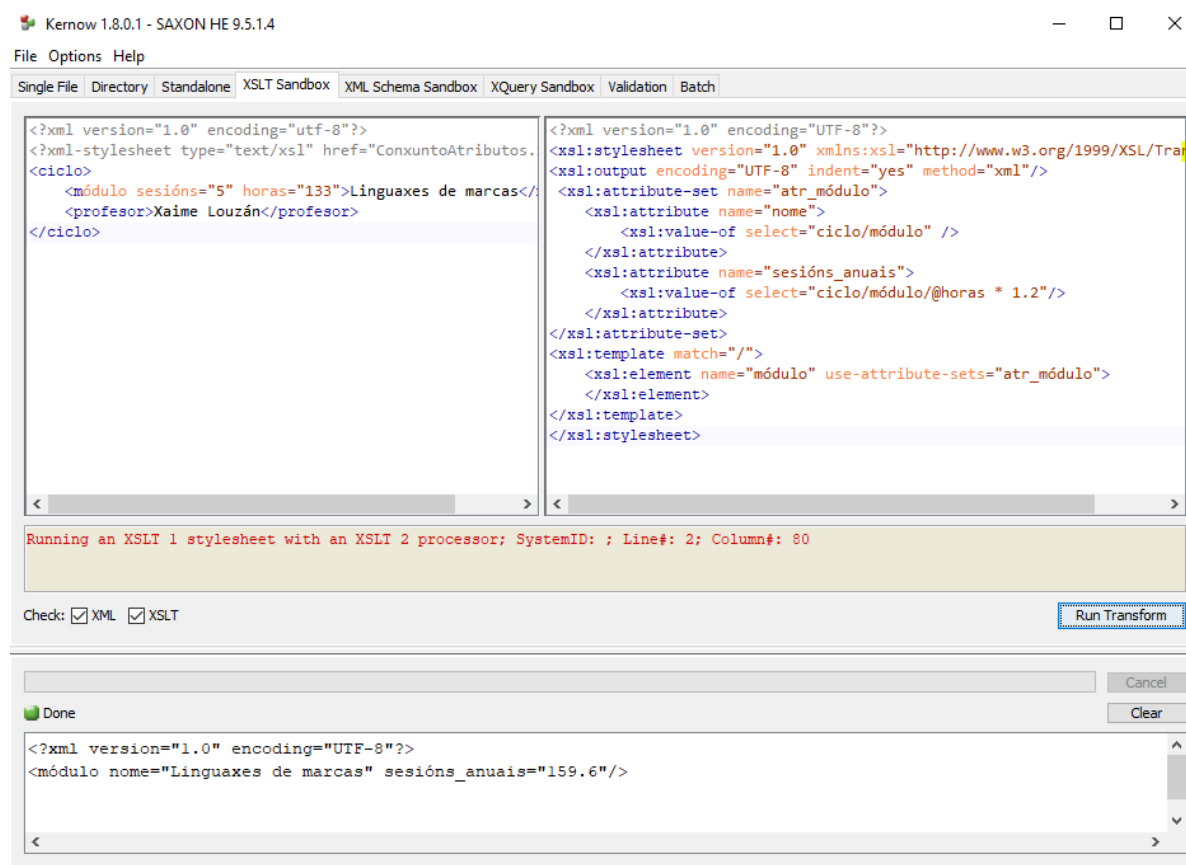
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="UTF-8" indent="yes" method="xml"/>
  <xsl:attribute-set name="atr_módulo">
    <xsl:attribute name="nome">
      <xsl:value-of select="ciclo/módulo" />
    </xsl:attribute>
    <xsl:attribute name="sesións_anuais">
      <xsl:value-of select="ciclo/módulo/@horas * 1.2"/>
    </xsl:attribute>
  </xsl:attribute-set>
```

```

<xsl:template match="/">
    <xsl:element name="módulo" use-attribute-sets="atr_módulo">
        </xsl:element>
    </xsl:template>
</xsl:stylesheet>

```

Obteríamos como resultado:



Creación de comentarios e instrucións de procesamento

Comentarios

Para crear un comentario no documento de saída teremos que empregar "`<xsl:comment>`". Por exemplo, poñendo:

```
<xsl:comment>Primeiro exemplo</xsl:comment>
```

Obteremos:

```
<!--Primeiro exemplo-->
```

Instrucións de procesamento

Engádense empregando "`<xsl:processing-instruction>`" e indicando obrigatoriamente o seu atributo "name".

Por exemplo, poderíamos asociar o documento XML resultante cun documento XSLT facendo:

```
<?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:processing-instruction name="xml-stylesheet">href="profes.xml"
type="text/xml"</xsl:processing-instruction>
  <!--
</xsl:template>
</xsl:stylesheet>
```

O resultado obtido é:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="profes.xml" type="text/xml"?>
...
```

Documentos XSLT con varios patrones

Xa vimos o que ocorre cando non existe un patrón para algún elemento do documento XML orixe. E ata o de agora estivemos a empregar documentos XSLT cun único patrón. Imos ver que ocorre cando temos un documento XSLT con varios patrones, como o seguinte **exemplo**.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/módulo/profesor">
  <Profesor />
</xsl:template>
<xsl:template match="/módulo">
  <Módulo />
</xsl:template>
</xsl:stylesheet>
```

Cando aplicamos a anterior transformación ao documento XML.

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

Obtemos como saída.

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

¡¡OLLO!! O primeiro patrón non chega a procesarse.

Cal é a explicación a este comportamento?

O motivo é que o **procesador XSLT vai collendo os nodos do documento orixinal (o .xml) de un en un, comezando co elemento raíz, e buscando algún patrón que se lle poida aplicar.**

Polo tanto o elemento raíz do documento orixe, "<módulo>", é o primeiro en ser procesado, e con él tódolos seus fillos. O patrón seleccionado polo procesador XSLT para este elemento é "<xsl:template match="/módulo">", e como resultado cópiase "<Módulo

/>" no documento de saída.

Unha vez procesados o elemento raíz xunto cos seus fillos, xa non quedan máis nodos por procesar no documento orixinal, co cal o patrón "<xsl:template match="/módulo/profesor">" non chega a executarse.

Procesar un elemento e os seus fillos

Existe unha forma de **facér que no procesamento dun elemento se procesen tamén os patróns correspondentes aos seus fillos**. Isto faise empregando "<xsl:apply-templates>" dentro dun patrón.

- *xsl:apply-templates* fai que se apliquen as regras que siguen a todos os nodos seleccionados.

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

- Pode restrinxirse co atributo *select* para especificar un subconxunto de nodos.

```
<xsl:apply-templates select="PATRÓN" />
```

Por **exemplo**, se aplicásemos a seguinte transformación ao documento anterior.

```
<?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="text"/>
  <xsl:template match="/">
    <xsl:apply-templates />
  </xsl:template>
</xsl:stylesheet>
```

Ao elemento "profesor" aplícalle o patrón predefinido, e polo tanto copia o texto que contén na saída, obtendo:

Xaime Louzán

En ocasións non queremos que se procesen todos os fillos dun elemento. Nestes casos podemos empregar o atributo "select" de "<xsl:apply-templates>" para indicar a expresión XPath que se corresponda con aqueles nodos que queremos que se procesen. Por **exemplo**:

```
<xsl:apply-templates select="profesor" />
```

Neste caso, se o elemento "<módulo>" tivera varios fillos, soamente continuaría o procesamento de "<profesor>".

Outro **exemplo**; no caso anterior poderíamos modificar o documento XSLT poñendo:

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

```

<xsl:template match="profesor">
  <Profesor />
</xsl:template>
<xsl:template match="/módulo">
  <xsl:element name="Módulo">
    <xsl:apply-templates select="profesor" />
  </xsl:element>
</xsl:template>
</xsl:stylesheet>

```

¡¡OLLO!! Neste caso, ao procesar o elemento raíz "<módulo>", crea o elemento "<Módulo>" no documento de saída, e despois busca algún patrón adecuado para procesar ao elemento "<profesor>", o que fai que se execute o patrón "<xsl:template match='profesor'>". O resultado sería:

```

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

```

É moi habitual atopar documentos XSLT coa seguinte estrutura:

```

<?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:apply-templates />
    ...
  </xsl:template>

  <xsl:template match="nodos1">
    ...
    <xsl:apply-templates />
    ...
  </xsl:template>

  <xsl:template match="nodos2">
    ...
    <xsl:apply-templates />
    ...
  </xsl:template>
</xsl:stylesheet>

```



Tarefa 3. Transformacións con varios patróns

Tomando como base o seguinte documento XML (*Actividad3.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 3_a

Obter un listado en formato XML dos discos que figuran no documento XML orixe, engadíndolles un atributo co nome da máquina no que se atopan. Por exemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<discos>
  <disco tecnoloxía="SATA" capacidade="2000" máquina="PC017"/>
  <disco tecnoloxía="SCSI" capacidade="200" máquina="GALILEO"/>
  <disco tecnoloxía="SCSI" capacidade="200" máquina="GALILEO"/>
  <disco tecnoloxía="SCSI" capacidade="200" máquina="GALILEO"/>
</discos>
```

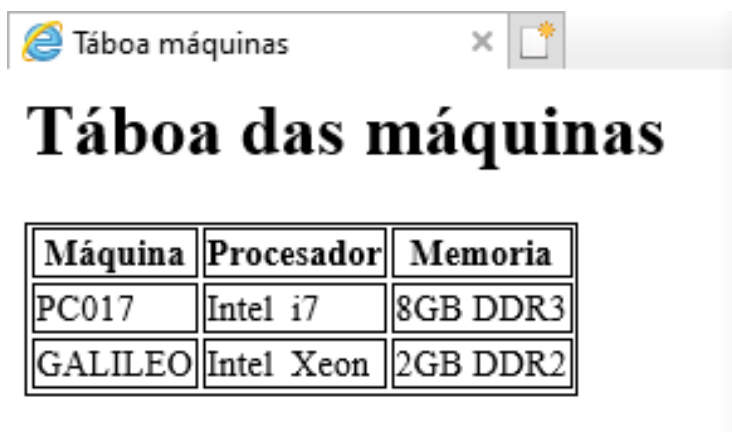
Intentade facelo vós, pero por se non vos sae, os deixo a solución.

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" indent="yes" method="xml"/>
  <xsl:template match="equipos">
    <discos>
      <xsl:apply-templates select="máquina/hardware/disco"/>
    </discos>
  </xsl:template>
  <xsl:template match="disco">
    <xsl:element name="disco">
      <xsl:attribute name="tecnoloxía">
        <xsl:value-of select="@tecnoloxía" />
      </xsl:attribute>
      <xsl:attribute name="capacidade">
        <xsl:value-of select="@capacidade" />
      </xsl:attribute>
      <xsl:attribute name="máquina">
        <xsl:value-of select="../../@nome" />
      </xsl:attribute>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

b) Tarefa 3_b

Obter un documento HTML como o seguinte como saída da transformación.



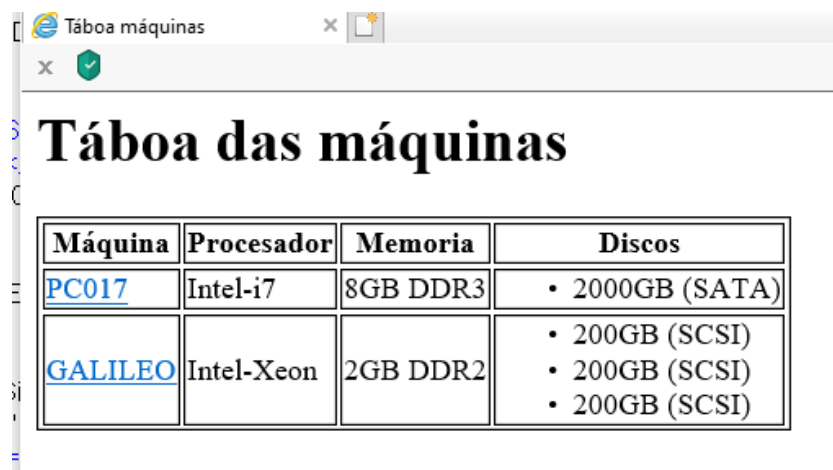
Cando definiades as etiquetas HTML dentro do XSLT, na cabeceira podedes poñer os estilos da táboa.

```
<head>
  <title>Táboa máquinas</title>
  <style>
    table, tr, th, td {border: 1px solid black;}
  </style>
</head>
```

c) Tarefa 3_c

Engadirlle á páxina web xerada na transformación anterior, un enlace na primeira columna de xeito que premendo no nome da máquina se abra a páxina web coa URL correspondente á dirección IP da máquina; esta IP se recolle do documento XML (por exemplo, <http://192.168.20.105>). Será unha simulación, pois estas direccións IPs non corresponden con ningún sitio web.

Engadir tamén unha nova columna ao final, na que figure para cada máquina unha lista non numerada dos seus discos, tal e como figura na imaxe.



Máquina	Procesador	Memoria	Discos
PC017	Intel-i7	8GB DDR3	• 2000GB (SATA)
GALILEO	Intel-Xeon	2GB DDR2	• 200GB (SCSI) • 200GB (SCSI) • 200GB (SCSI)

d) Tarefa 3_d

Transformar o documento orixe noutro documento XML co seguinte formato.

```
<?xml version="1.0" encoding="UTF-8"?>
<máquinas>
  <máquina tipo="PC-Sobremesa" fabricante="Dell" procesador="Intel-4-núcleos-a-3.1-GHz" discos="1-con-2000GB"/>
  <máquina tipo="Torre" fabricante="Fujitsu-Siemens" procesador="Intel-4-núcleos-a-3-GHz" discos="3-con-600GB"/>
</máquinas>
```

Empregar un conxunto de atributos na especificación de transformación.

Intentade facelo vós, pero por se non vos sae, os deixo a solución.

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" indent="yes" method="xml"/>
  <xsl:attribute-set name="atr_máquina">
    <xsl:attribute name="tipo">
      <xsl:value-of select="hardware/tipo"/>
    </xsl:attribute>
    <xsl:attribute name="fabricante">
      <xsl:value-of select="hardware/fabricante"/>
    </xsl:attribute>
```

```

    <xsl:attribute name="procesador">
      <xsl:value-of select="concat (hardware/procesador/@marca, ' ', hardware/
re/procesador/@num_nucleos, ' núcleos a ', hardware/procesador/@velocidade,
' GHz') " />
    </xsl:attribute>
    <xsl:attribute name="discos">
      <xsl:value-of select="concat (count (hardware/disco), ' con ',
sum (hardware/disco/@capacidade), 'GB') " />
    </xsl:attribute>
  </xsl:attribute-set>
  <xsl:template match="equipos">
    <xsl:element name="máquinas">
      <xsl:apply-templates select="máquina"/>
    </xsl:element>
  </xsl:template>
  <xsl:template match="máquina">
    <xsl:element name="máquina" use-attribute-sets="atr_máquina" />
  </xsl:template>
</xsl:stylesheet>

```



Tarefa 4. Transformacións con varios patróns (II)

Tomando como base o documento XML *Actividad4.xml*, que ten o horario dun ciclo de DAW ordinario (podedes adaptalo co voso horario) imos facer as seguintes tarefas.

a) Tarefa 4_a

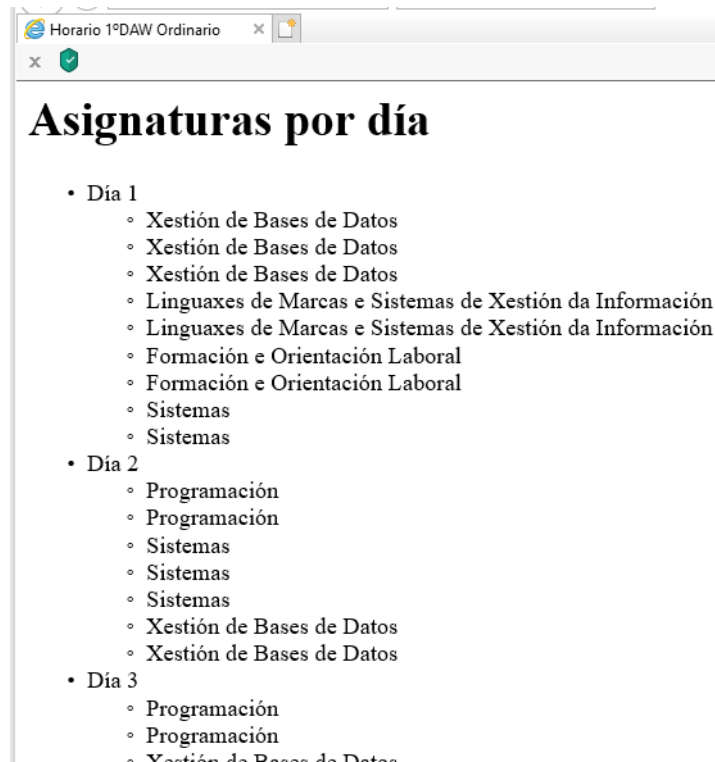
Queremos obter como saída unha páxina web coa seguinte información:

Horario 1ºDAW, curso 2020/21

	Inicio	Fin
1 Hora	08:45	09:35
2 Hora	09:35	10:25
3 Hora	10:25	11:15
4 Hora	11:15	12:05
5 Hora	12:05	12:55
6 Hora	12:55	13:45
7 Hora	13:45	14:35
8 Hora	16:20	17:10
9 Hora	17:10	18:00

b) Tarefa 4_b

Empregando como orixe o mesmo documento XML da tarefa anterior, obter o seguinte documento HTML como saída da transformación:



c) Tarefa 4_c

Tomando como orixe o mesmo documento XML da tarefa anterior, obter como saída da transformación un documento XML con raíz "<materias>", que soamente conteña a lista dos días coas materias tal e como figuran no documento orixe, pero non o elemento "<horas>" cos seus fillos "<hora>". Isto é, queremos obter o seguinte:

```
<?xml version="1.0" encoding="UTF-8"?>
<materias>
  <dia-num="1">
    <materia.hora="1".nome="Xestión de Bases de Datos"/>
    <materia.hora="2".nome="Xestión de Bases de Datos"/>
    <materia.hora="3".nome="Xestión de Bases de Datos"/>
    <materia.hora="4".nome="Linguaxes de Marcas e Sistemas de Xestión da Información"/>
    <materia.hora="5".nome="Linguaxes de Marcas e Sistemas de Xestión da Información"/>
    <materia.hora="6".nome="Formación e Orientación Laboral"/>
    <materia.hora="7".nome="Formación e Orientación Laboral"/>
    <materia.hora="8".nome="Sistemas"/>
    <materia.hora="9".nome="Sistemas"/>
  </dia>
  <dia-num="2">
    <materia.hora="1".nome="Programación"/>
    <materia.hora="2".nome="Programación"/>
    <materia.hora="3".nome="Sistemas"/>
    <materia.hora="4".nome="Sistemas"/>
    <materia.hora="5".nome="Sistemas"/>
    <materia.hora="6".nome="Xestión de Bases de Datos"/>
    <materia.hora="7".nome="Xestión de Bases de Datos"/>
  </dia>
  <dia-num="3">
    <materia.hora="1".nome="Programación"/>
    <materia.hora="2".nome="Programación"/>
    <materia.hora="3".nome="Xestión de Bases de Datos"/>
  </dia>
</materias>
```