



LMSGI06.1 Transformación de datos



☰ Transformación de datos

☰ Estructura básica

☰ Instrucciones

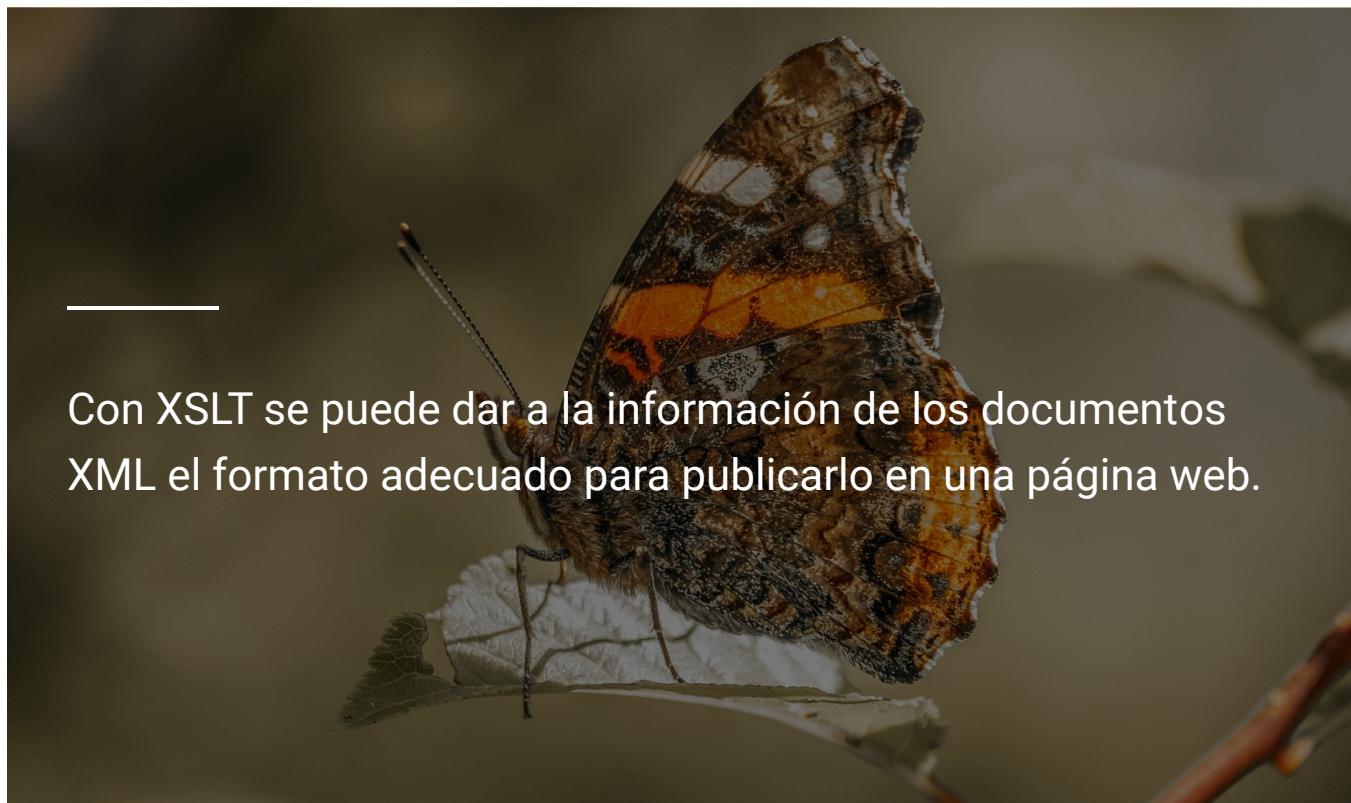
?

Autoevaluación

CONTENIDO PDF

☰ Contenido PDF

Transformación de datos



Con XSLT se puede dar a la información de los documentos XML el formato adecuado para publicarlo en una página web.

XSLT

Los documentos XML son documentos de texto con etiquetas que contienen exclusivamente información sin entrar en detalles de formato. Esto implica la necesidad de algún medio para expresar la transformación de un documento XML, para que una persona pueda utilizar directamente los datos para leer, imprimir, etc.

XSL es una familia de lenguajes que sirven para definir transformaciones y presentaciones de documentos XML. Las tecnologías que entran en juego en la transformación de documentos son:

- XSLT: permite definir el modo de transformar un documento XML en otro.
- XSL-FO: se utiliza para transformar XML en un formato legible e imprimible por una persona, por ejemplo en un documento PDF.
- XPath: permite el acceso a los diversos componentes de un documento XML

XSLT (Transformaciones XSL) es un lenguaje de programación declarativo y un [estándar](#) aprobado por el W3C que permite generar documentos en diferentes formatos a partir de documentos XML.

En este punto, ¿una hoja XSLT es también un documento XML? Sí, XSLT es uno de los lenguajes derivados de XML, por tanto las hojas XSLT también son documentos XML (al igual que sucede con los canales RSS, atom o los documentos XSD).

¿Qué transformaciones podemos realizar sobre un documento XML usando XSLT?

- A otro documento XML.
- A un documento HTML.
- A un documento de texto.

Una de las limitaciones de XSLT es que el documento de entrada debe ser siempre XML.

XML y editores de texto

NOTE PAD++

COPY EDITOR

NOTE PAD++

COPY EDITOR

Proceso

El proceso a seguir sería el siguiente:

1

El documento XML es el documento inicial al que se aplican las transformaciones.

2

La hoja de estilo XSLT es el documento que contiene las reglas de transformación que se van a aplicar al documento inicial XML. Estas hojas de estilo son también archivos XML.

3

El procesador XSLT es el programa que aplica al documento inicial XML las reglas de transformación incluidas en la hoja de estilo XSLT y genera el documento final.

El resultado de la ejecución del programa es un nuevo documento que puede ser un documento XML ó no.

Herramientas

Para la transformación de un documento XML podemos:

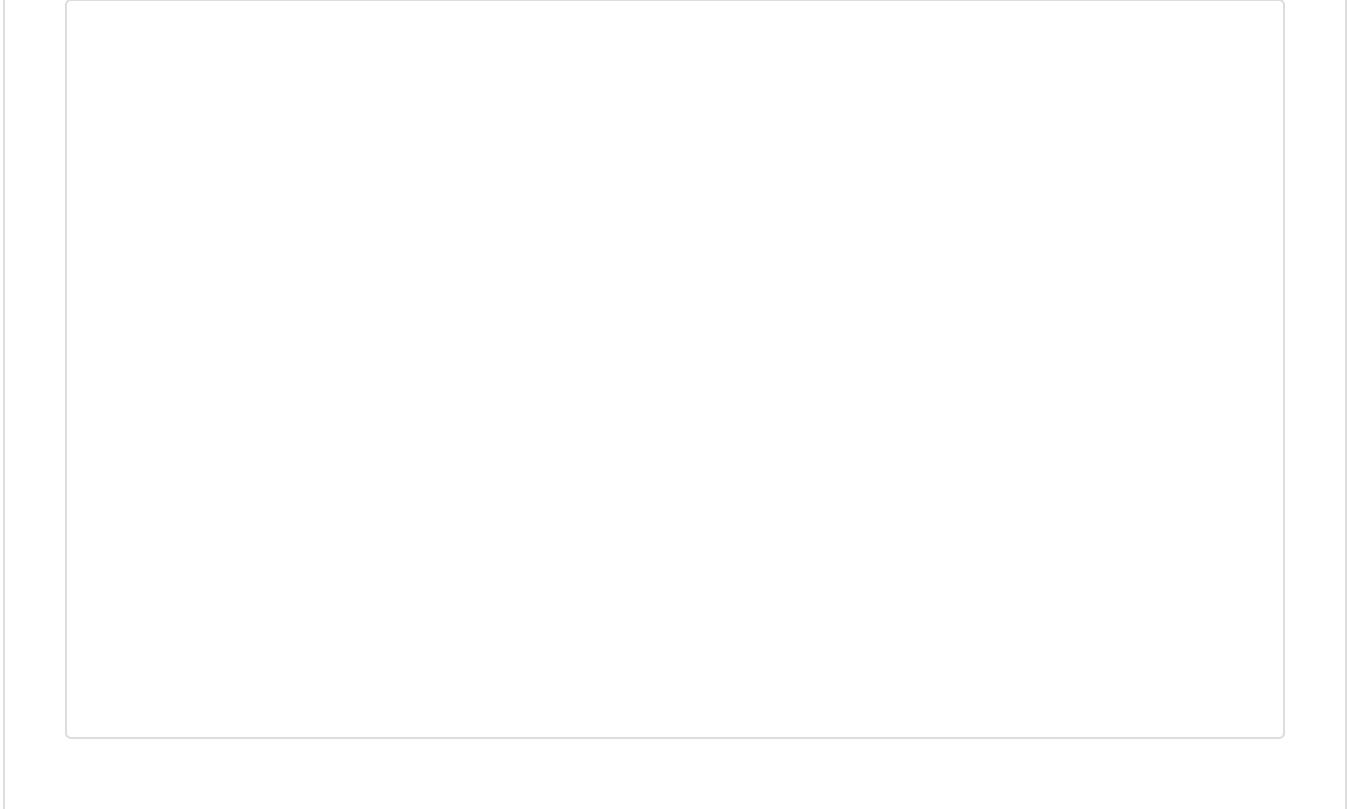
- **Utilizar un servidor web:** es la **mejor opción**. En este caso es necesario poner en marcha un servidor web y un editor de texto plano que puede ser Notepad++ ó XML Copy Editor. Con la ejecución del documento xml, se muestra en una página web el resultado final de la transformación, no el documento intermedio que habría que mostrar en un navegador, como en el caso de no utilizar un servidor web.
- **No utilizar un servidor web:** en cuyo caso con el editor XML Copy Editor ó Notepad++ sería suficiente. Es la opción más rápida y sencilla, pero hay que tener mucho cuidado con los **caracteres raros** (por ejemplo: nbsp) que puedan existir en el archivo xsl, siendo necesario eliminarlos. Tras la transformación, es posible utilizar navegadores web para ver el resultado final.

Formas de ejecutar XSLT

WEB SERVER

COPY EDITOR

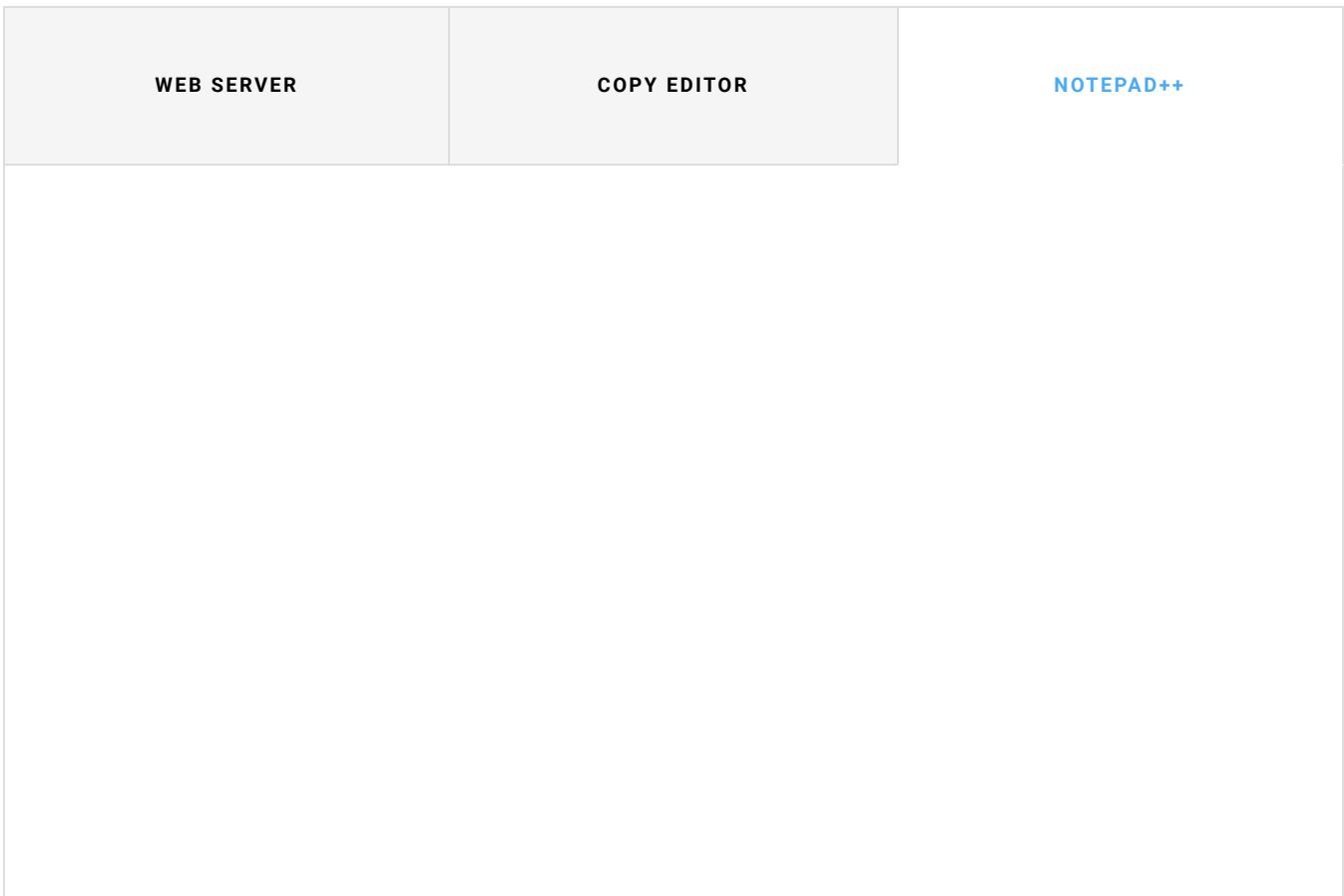
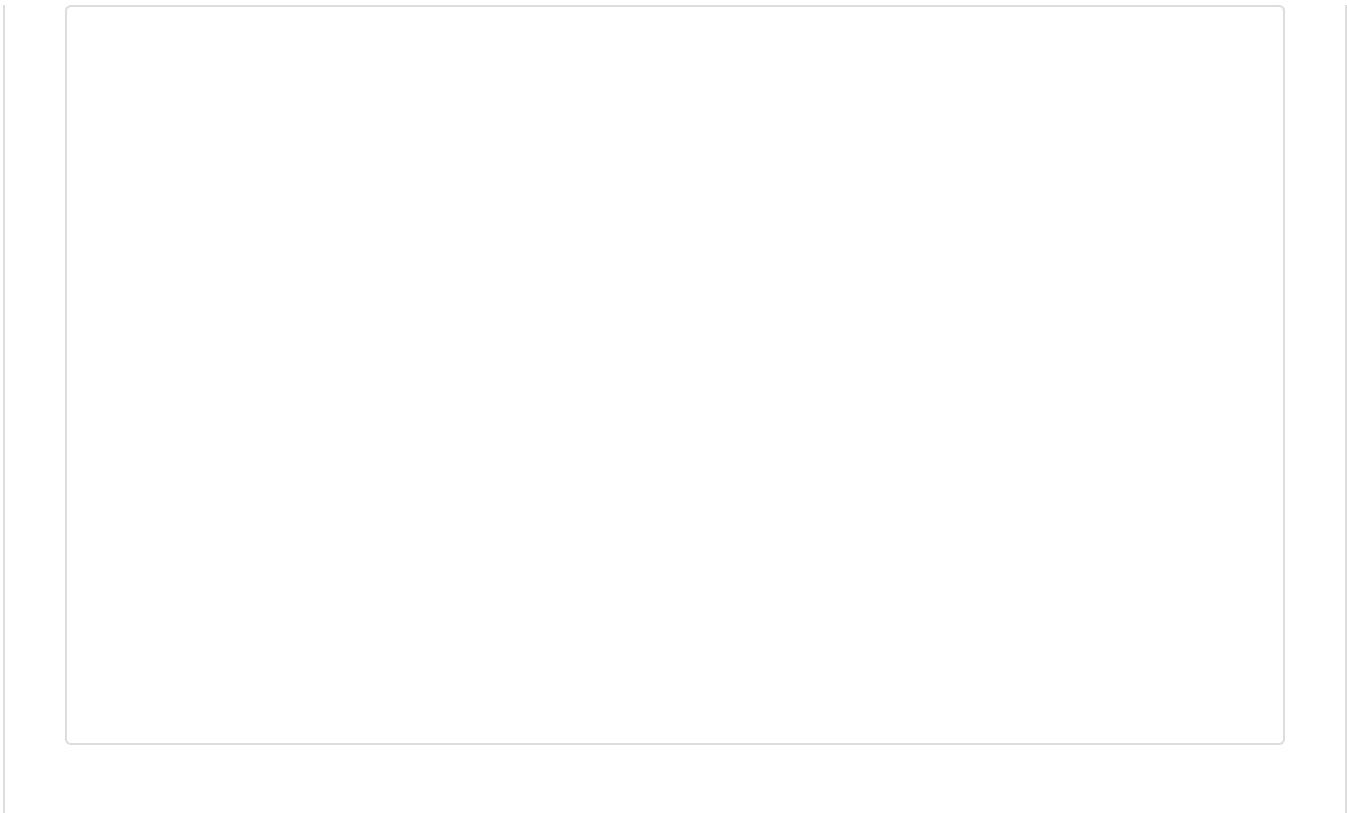
NOTE PAD++



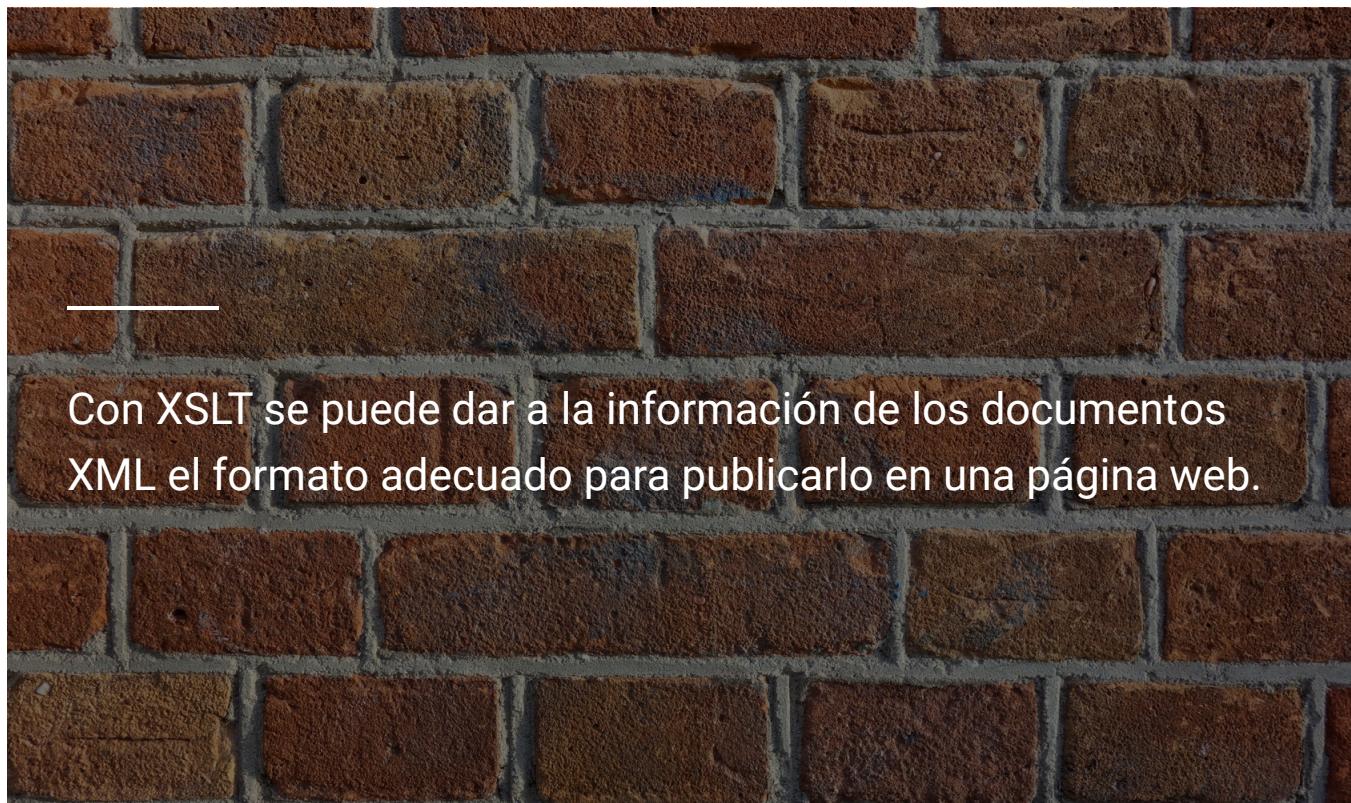
WEB SERVER

COPY EDITOR

NOTE PAD++



Estructura básica



Con XSLT se puede dar a la información de los documentos XML el formato adecuado para publicarlo en una página web.

Estructura básica

Una hoja de estilo ó plantilla XSLT es también un documento XML y debe estar bien formado.

Las hojas de estilo se guardan siempre en archivos independientes con extensión ".xsl" y deben empezar con la declaración XML:

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

Los elementos básicos que puede contener una plantilla son los siguientes:

- **Elementos XSLT:** están precedidos por el prefijo "xsl:" porque pertenecen al espacio de nombres xsl. Están definidos en el estándar del lenguaje y son interpretados por cualquier procesador XSLT.
- **Elementos literales:** son cualquier elemento que no sea instrucción y no empiece por "xsl:". No pertenecen al lenguaje sino que se repiten en la salida sin más.

El elemento raíz de la hoja de est XSLT es stylesheet. Este elemento contendrá todos los demás y debe ir precedido por el prefijo "xsl:" y el correspondiente espacio de nombres para hojas de estilo XSLT. Los nombres de los elementos XSLT proceden de un mismo espacio de nombres y deben escribirse precedidos por el preijo xmlns:xsl="http://www.w3.org/1999/XSL/Transform".

El elemento raíz queda así:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <!-- c ó d i g o: reglas de transformación -->
</xsl:stylesheet>
```

Entre las marcas de inicio y fin del elemento raíz (xsl:stylesheet) se escriben las **reglas de transformación** definidas mediante el elemento **xsl:template**. Cada regla indica:

- qué instancias de los elementos del documento XML se van a transformar
- y cómo lo hará cada una de ellas.

Ejemplos

Ejemplo 1

En este ejemplo, la regla se aplicará a todas las instancias del elemento **libro**. Esto se indica mediante el atributo `match` que acompaña al elemento **xsl:template**.

Entre las etiquetas de inicio y fin del elemento **xsl:template** se escribe la transformación que se debe realizar. Es decir, qué texto y qué marcas se escribirán en el documento resultado de la transformación cada vez que se encuentre una instancia del elemento **libro** en el documento origen. En concreto, `<xsl:value-of>` recupera y escribe en el documento resultado el valor del elemento **autor** que está siendo procesado.



Ejemplo1.xml

360 B



Ejemplo1.xsl

222 B



ARCHIVO XML

ARCHIVO XSL

RESULTADO

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="Ejemplo1.xsl"?>
<libros>
  <libro>
    <titulo>El Principito</titulo>
    <autor>Antoine de Saint-Exupéry</autor>
    <anyo>1943</anyo>
  </libro>
  <libro>
    <titulo>1984</titulo>
    <autor>George Orwell</autor>
    <anyo>1949</anyo>
  </libro>
</libros>
```

ARCHIVO XML	ARCHIVO XSL	RESULTADO
	<pre><?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:template match="//libro"> <xsl:value-of select="autor"/> </xsl:template> </xsl:stylesheet></pre>	

ARCHIVO XML	ARCHIVO XSL	RESULTADO
		El resultado en XML Copy Editor se vería así:

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

Antoine de Saint-Exupéry
George Orwell
```

Ejemplo 2

En este ejemplo, que es el caso de que no existe plantilla, no hay reglas para aplicar por lo que el procesador recorre todos los nodos y extrae el texto contenido en cada nodo.



Ejemplo2.xml

360 B



Ejemplo2.xsl

141 B



ARCHIVO XML

ARCHIVO XSL

RESULTADO

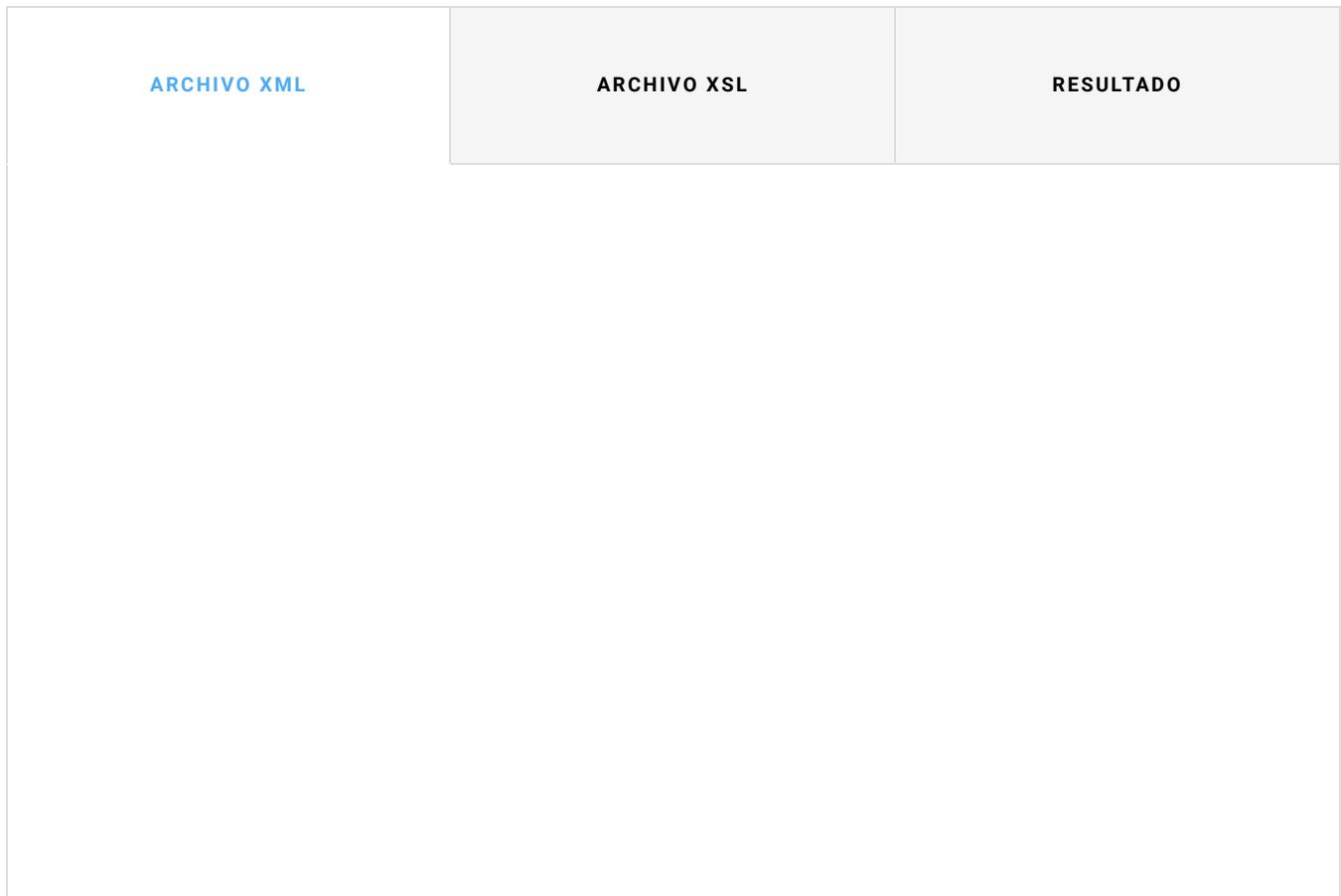
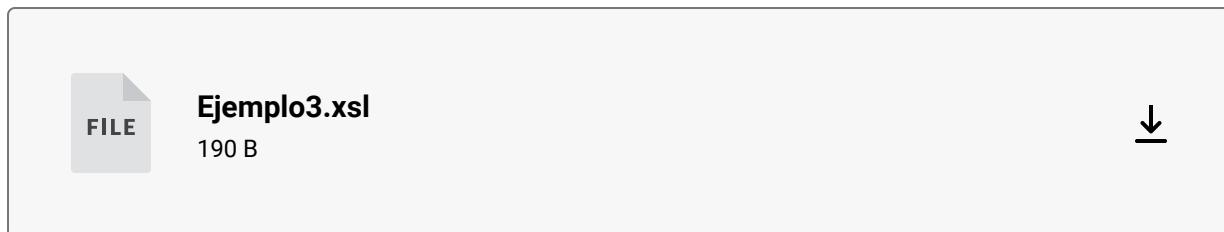
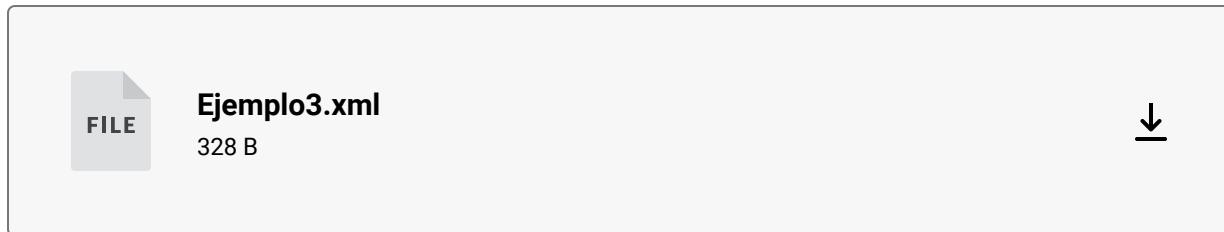
```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="Ejemplo2.xsl"?>
<libros>
  <libro>
    <titulo>El Principito</titulo>
    <autor>Antoine de Saint-Exupéry</autor>
    <anyo>1943</anyo>
  </libro>
  <libro>
    <titulo>1984</titulo>
    <autor>George Orwell</autor>
    <anyo>1949</anyo>
  </libro>
</libros>
```

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

ARCHIVO XML	ARCHIVO XSL	RESULTADO
El resultado en XML Copy Editor se vería así:		
<?xml version="1.0" encoding="UTF-8"?> ` El Principito Antoine de Saint-Exupéry 1943 1984 George Orwell 1949		

Ejemplo 3

En este ejemplo, que es el caso de que la plantilla está vacía, el procesador no genera ningún resultado en el documento final ni recorre los nodos hijos.



```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="Ejemplo3.xsl"?>
<libros>
<libro>
<titulo>El Principito</titulo>
<autor>Antoine de Saint-Exupéry</autor>
<anyo>1943</anyo>
</libro>
<libro>
<titulo>1984</titulo>
<autor>George Orwell</autor>
<anyo>1949</anyo>
</libro>
</libros>
```

ARCHIVO XML

ARCHIVO XSL

RESULTADO

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

ARCHIVO XML

ARCHIVO XSL

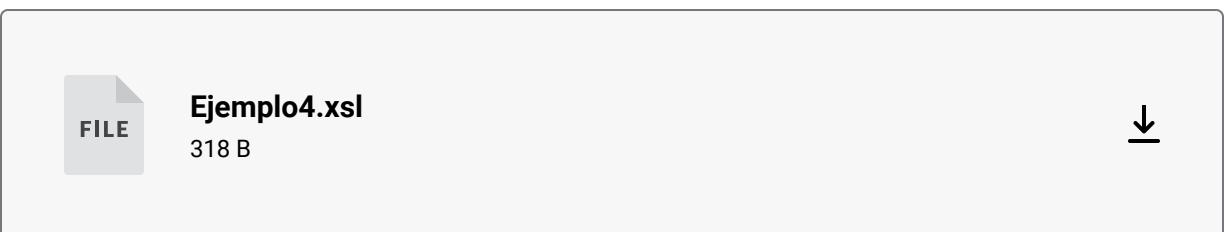
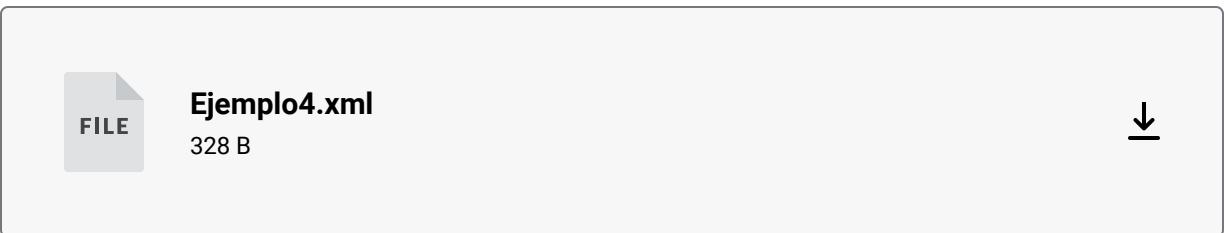
RESULTADO

El resultado en XML Copy Editor se vería así:

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

Ejemplo 4

Cuando existe más de una plantilla en el documento XSL hay que tener en cuenta que, si ambas actúan sobre el mismo nodo, el procesador solo aplica la última regla a cada nodo. Así, en este caso obtendríamos el contenido de los elementos autor.



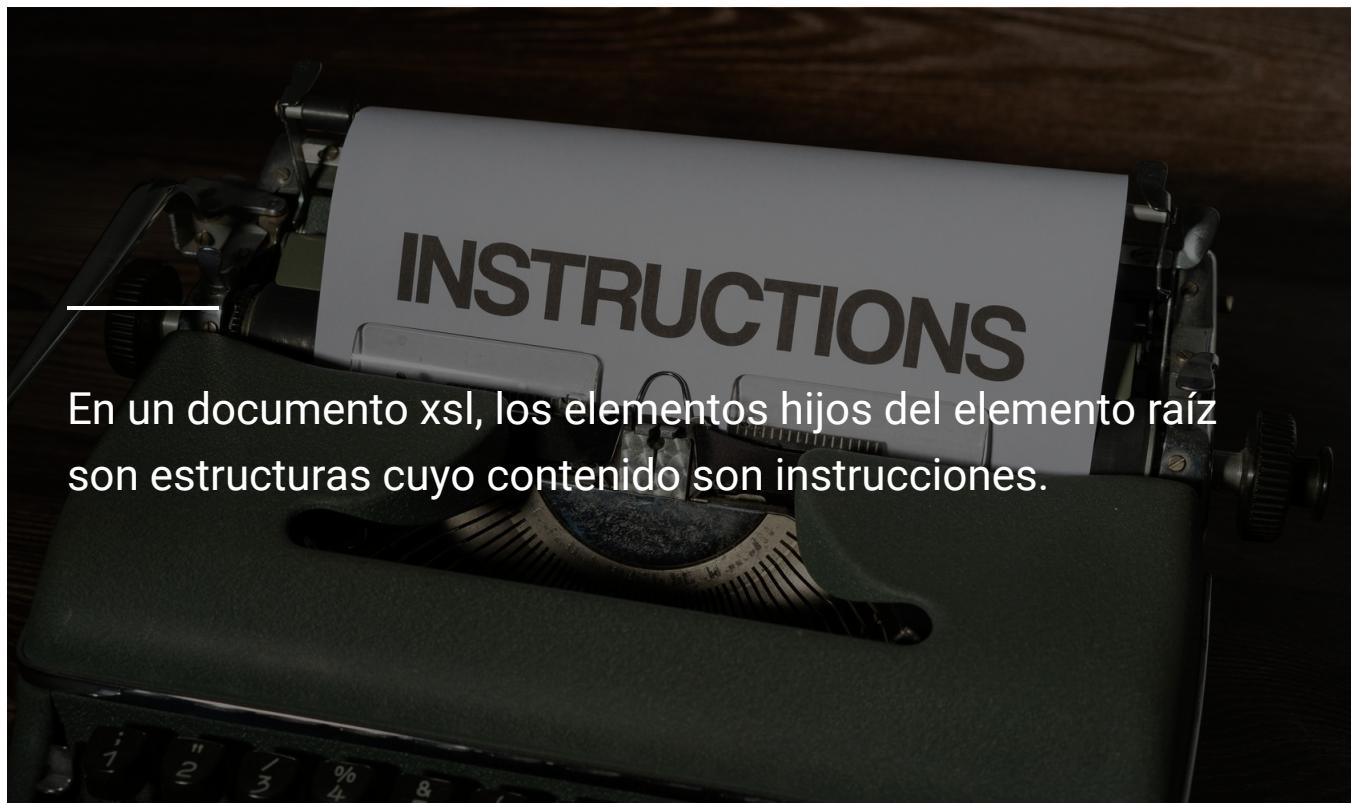
ARCHIVO XML	ARCHIVO XSL	RESULTADO

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="Ejemplo4.xsl"?>
<libros>
<libro>
<titulo>El Principito</titulo>
<autor>Antoine de Saint-Exupéry</autor>
<anyo>1943</anyo>
</libro>
<libro>
<titulo>1984</titulo>
<autor>George Orwell</autor>
<anyo>1949</anyo>
</libro>
</libros>
```

ARCHIVO XML	ARCHIVO XSL	RESULTADO
	<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:template match="//libro"> <p><xsl:value-of select="titulo"/></p> </xsl:template> <xsl:template match="//libro"> <p><xsl:value-of select="autor"/></p> </xsl:template> </xsl:stylesheet>	

ARCHIVO XML	ARCHIVO XSL	RESULTADO
		<p>El resultado en XML Copy Editor se vería así:</p> <pre><?xml version="1.0" encoding="UTF-8"?> <p>Antoine de Saint-Exupéry</p> <p>George Orwell</p></pre>

Instrucciones



En un documento xsl, los elementos hijos del elemento raíz son estructuras cuyo contenido son instrucciones.

Instrucciones

Ya se han visto algunas pero entre las instrucciones más importantes:

xsl:value-of

—

Calcula el valor de una expresión XPath dada y lo inserta en el árbol de resultados del documento de salida. Además, en XSLT podemos 'filtrar' ó indicar qué instancias de un elemento queremos procesar en base al valor.

Por ejemplo: para obtener autores que publicaron algún libro en el año 1943.

___ Archivo xml:

El del ejemplo anterior

___ Archivo 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="//libro">
<xsl:value-of select="autor[../anyo='1943']"/>
</xsl:template>
</xsl:stylesheet>
```

En este ejemplo, se aplica una plantilla a los elementos libro y se obtienen los autores de los libros que se publicaron en el año 1943.

En XML Copy Editor el resultado sería:

```
<?xml version="1.0" encoding="UTF-8"?>
Antoine de Saint-Exupéry
```

xsl:attribute

—

Añade un atributo en el documento resultado de la transformación.

Por ejemplo:

___ Archivo xml:

El del ejemplo anterior

___ Archivo 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="//libro">
<p><artista>
<xsl:attribute name="año">
```

```
<xsl:value-of select="anyo"/>
</xsl:attribute>
<xsl:value-of select="autor[./anyo='1943']"/>
</artista></p>
</xsl:template>
</xsl:stylesheet>
```

En este ejemplo, se aplica una plantilla a los elementos libro se crean unas etiquetas y un atributo con el valor del año en el resultado y el autor si el libro ha sido publicado en el año 1943.

En XML Copy Editor el resultado sería:

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

<p>
  <artista año="1943">Antoine de Saint-Exupéry</artista>
</p>

<p>
  <artista año="1949"/>
</p>
```

xsl:if

—

Permite decidir si se va a procesar ó no una parte del documento XSL en función de una condición. Lleva un atributo test que contiene la condición. Si la condición se cumple para el elemento que se está procesando, la regla se ejecutará.

Por ejemplo:

___ Archivo xml:

El del ejemplo anterior

___ Archivo 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="//libro">
<xsl:if test="anyo='1949' ">
<xsl:value-of select="autor"/>
</xsl:if>
</xsl:template>
</xsl:stylesheet>
```

En este ejemplo, se aplica una plantilla a los elementos libro y se obtiene el autor de los que se han publicado en el año 1949.

En XML Copy Editor el resultado sería:

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

```
  George Orwell
```

xsl:choose, xsl:when, xsl:otherwise

—

Indican qué transformación se debe realizar en el caso de que se cumpla una condición. y en el resto de casos. Se utilizan de forma conjunta:

- El elemento xsl:choose contendrá uno o más elementos xsl:when y un elemento xsl:otherwise.
- Cada elemento xsl:when incluye un atributo test que tomará como valor la expresión a evaluar. Si se cumple, se ejecutará el código escrito entre las etiquetas de inicio y de fin del elemento xsl:when.
- El elemento xsl:otherwise contendrá el código que se ejecutará si no se cumple la condición del test.

Por ejemplo: para destacar en una tabla los elementos cuyo año de publicación es anterior a 1945.

___ Archivo xml:

El del ejemplo anterior

___ Archivo xsl:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="//libro">
<html>
<body>
<table border="1">
<tr>
<xsl:choose>
<xsl:when test="anyo < 1945">
<td bgcolor="#ff00ff"><xsl:value-of select="autor"/> </td>
</xsl:when>
<xsl:otherwise>
<td><xsl:value-of select="autor"/> </td>
</xsl:otherwise>
</xsl:choose>
</tr>
</table>
```

```
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

En este ejemplo, se aplica una plantilla a los elementos libro y se obtiene una tabla donde se marcan los autores de los libros publicados antes de 1945.

En XML Copy Editor el resultado sería:

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

<html>
  <body>
    <table border="1">
      <tr>
        <td bgcolor="#ff00ff">Antoine de Saint-Exupéry</td>
      </tr>
    </table>
  </body>
</html>

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

<html>
  <body>
    <table border="1">
      <tr>
        <td>George Orwell</td>
      </tr>
    </table>
  </body>
</html>
```

xsl:sort

Las reglas se ejecutan en el orden en el que se van encontrando los elementos en el documento. Para ordenar los contenidos se utiliza el elemento xsl:sort que puede llevar dos atributos:

- select: para indicar el nombre del elemento utilizado para la ordenación
- order: para indicar el orden de la ordenación.

— Archivo xml:
El del ejemplo anterior

___ Archivo xsl:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
  <xsl:for-each select="/libros/libro">
    <xsl:sort select="anyo" order="descending"/>
    <xsl:value-of select="autor"/>
    <xsl:text>&#10;</xsl:text> <!-- Se añade un salto de línea -->
  </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

En este ejemplo, se aplica una plantilla al elemento raíz y se realiza una iteración por cada libro que se ordena por año de publicación en orden descendente, y selecciona su autor.

En XML Copy Editor el resultado sería:

```
<?xml version="1.0" encoding="UTF-8"?>
George Orwell
Antoine de Saint-Exupéry
```

xsl:apply-templates

—

Se utiliza para aplicar una plantilla sobre un determinado elemento.

Por ejemplo:

___ Archivo xml:

El del ejemplo anterior

___ Archivo xsl:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
  <xsl:apply-templates/>
</xsl:template>
<xsl:template match="libro">
  <p>
    <xsl:apply-templates select="titulo"/>
    <xsl:apply-templates select="anyo"/>
  </p>
</xsl:template>
<xsl:template match="titulo">
  Título: <span style="color:#0000ff"><xsl:value-of select="."/></span><br/>
```

```
</xsl:template>
<xsl:template match="anyo">
Publicación: <span style="color:#ff0000"><xsl:value-of select="."/></span><br/>
</xsl:template>
</xsl:stylesheet>
```

En este ejemplo, nos situamos en el elemento raíz y aplicamos las plantillas. La primera plantilla se aplica sobre los elementos libro y se aplica una plantilla sobre los elementos título y otra plantilla sobre los elementos anyo.

En XML Copy Editor el resultado sería:

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

<p>
Título: <span style="color:#0000ff">El Principito</span><br/>
Publicación: <span style="color:#ff0000">1943</span><br/></p>

<p>
Título: <span style="color:#0000ff">1984</span><br/>
Publicación: <span style="color:#ff0000">1949</span><br/></p>
```

xsl:for-each

—

Permite hacer iteraciones sobre un conjunto de elementos.

Por ejemplo:

___ Archivo xml:

El del ejemplo anterior

___ Archivo 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="/">
<xsl:for-each select="/libros/libro">
    Título: <span style="color:#0000ff">
        <xsl:value-of select="titulo"/> </span><br />
    Publicación: <span style="color:#ff0000">
```

```
<xsl:value-of select="anyo"/> </span><br />
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

En este ejemplo, nos situamos en el elemento raíz y hacemos una iteración por cada elemento libro en el que se obtienen el título y el año de publicación de cada uno.

En XML Copy Editor el resultado sería:

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

Título:

```
<span style="color:#0000ff">El Principito</span>
<br/>
```

Publicación:

```
<span style="color:#ff0000">1943</span>
<br/>
```

Título:

```
<span style="color:#0000ff">1984</span>
<br/>
```

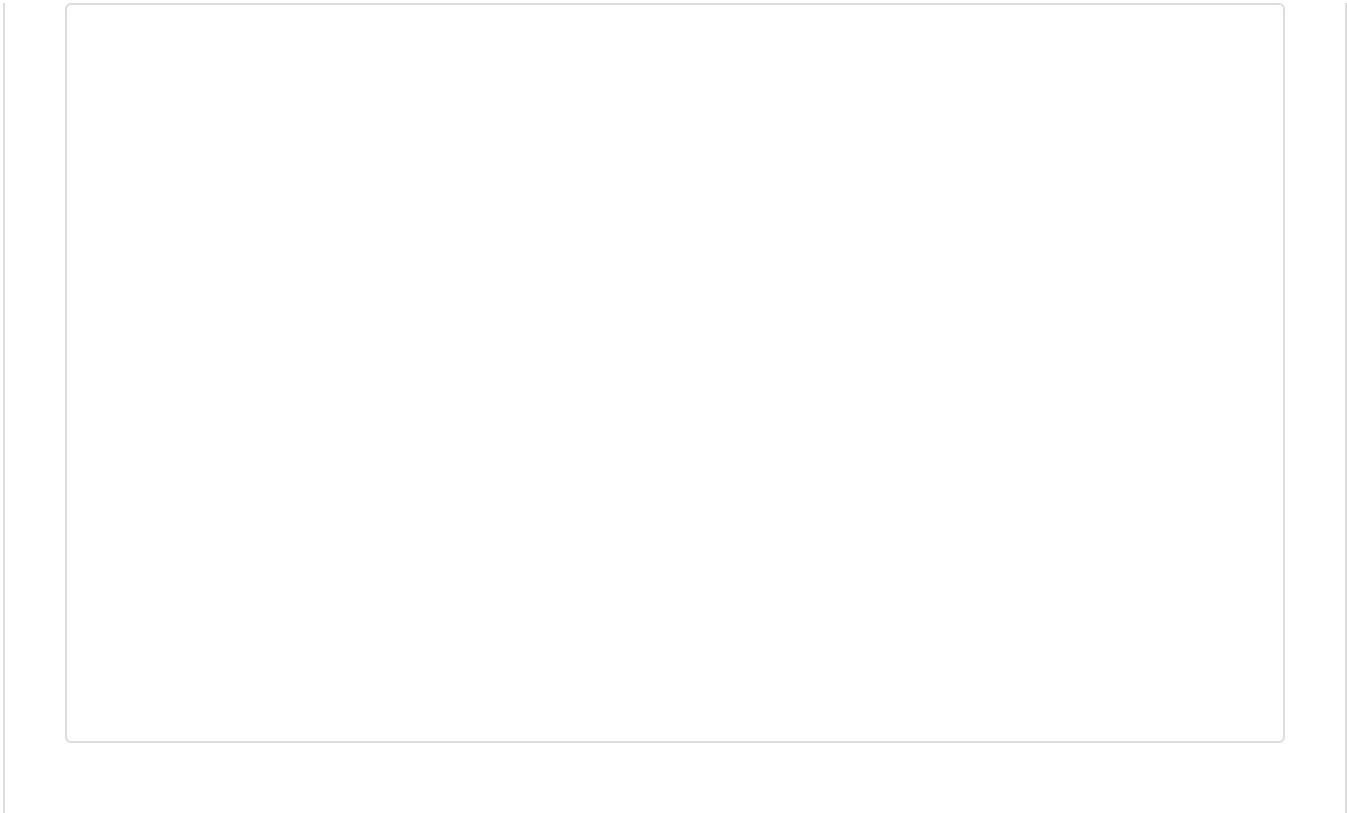
Publicación:

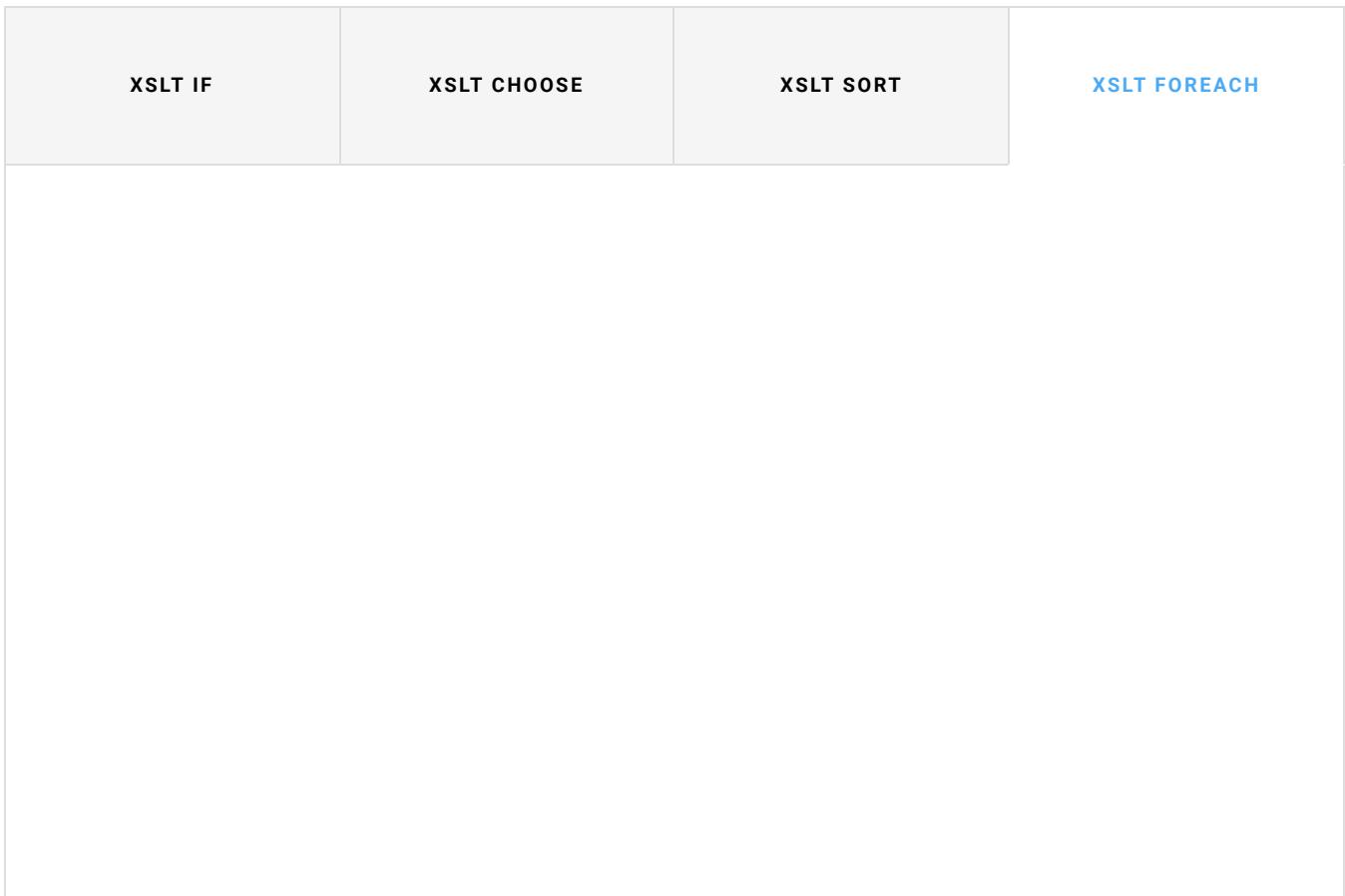
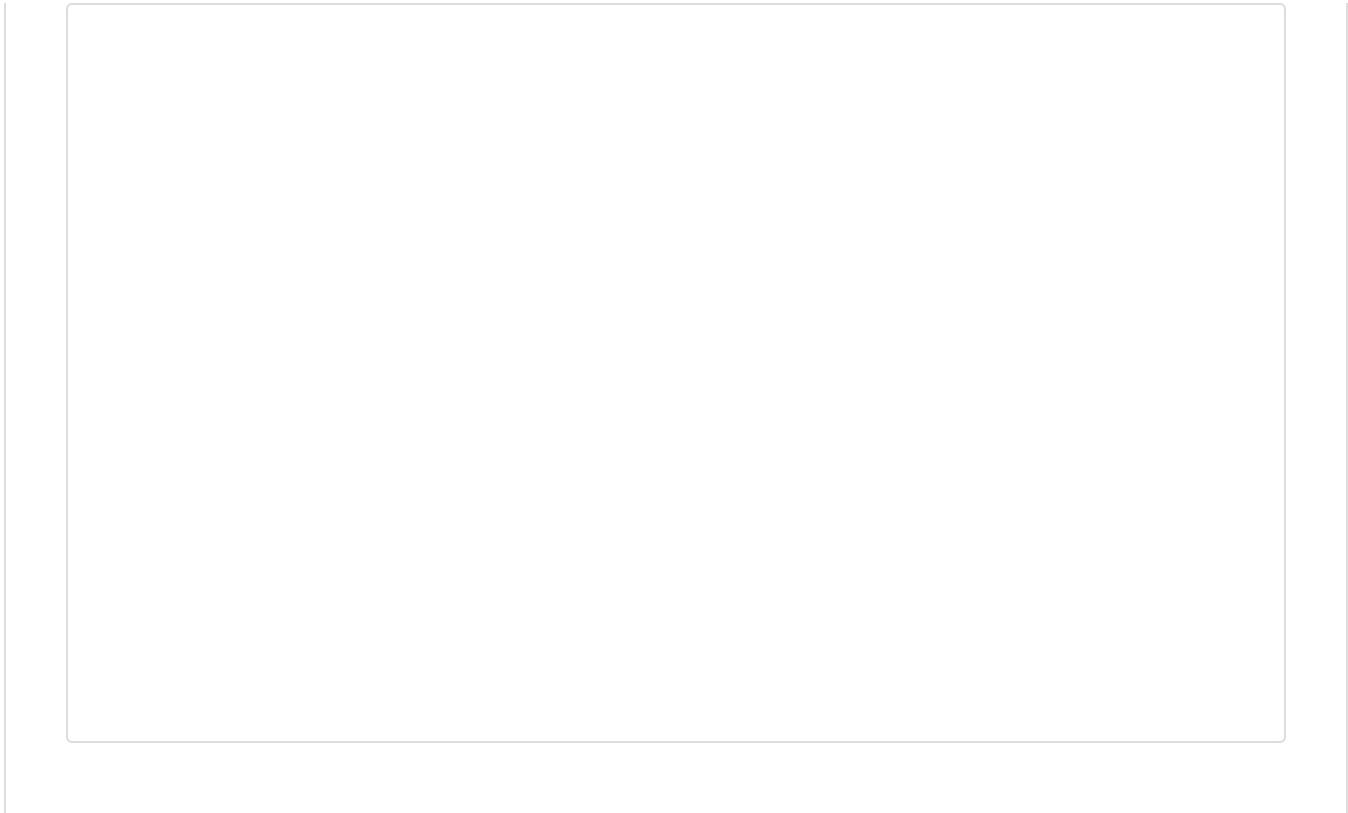
```
<span style="color:#ff0000">1949</span>
<br/>
```

VIDEOTUTORIALES

XSLT IF	XSLT CHOOSE	XSLT SORT	XSLT FOREACH

XSLT IF	XSLT CHOOSE	XSLT SORT	XSLT FOREACH







Ejemplos

EJEMPLO 1

EJEMPLO 2

EJEMPLO 1

EJEMPLO 2

Autoevaluación

Pregunta

01/03

XSL es una familia de lenguajes:

- Que sirven para definir transformaciones y presentaciones de documentos XML.
- Donde con XPath se permite el acceso a los diversos componentes de un documento XML
- Donde con XSLT se permite definir el modo de transformar un documento XML en otro.
- Ninguna de las demás opciones es verdadera.

Pregunta

02/03

Los elementos básicos que puede contener una plantilla XSLT son:

- Elementos XSLT que están precedidos por el prefijo "xsl:".
- Instrucciones FLWOR (For-Let-Where-Order by-Return).
- Todas las demás son verdaderas.
- Elementos literales que son cualquier elemento que no sea instrucción y no empiece por "xsl:".

Pregunta

03/03

Con "**xsl:value-of**":

- Se calcula el valor de una expresión XPath y lo inserta en el árbol de resultados del documento de salida.
- No es válido usarlo.
- Solamente se puede utilizar si se dispone de un esquema que lo valide.

Contenido PDF

Documento descargable en PDF



Este documento contiene una versión en PDF de las lecciones anteriores. Ten en cuenta que los elementos interactivos no funcionarán de la misma manera en este formato, por lo que se recomienda complementar su uso con la versión interactiva.

Contenido PDF

Puedes visualizar y descargar el contenido PDF a través de este enlace

IR AL ENLACE