LENGUAJE DE MARCAS

Y

SISTEMAS DE GESTION

DE LA INFORMACIÓN

RA5: Realiza conversiones sobre documentos para el intercambio de información

Tema 5: Conversión y adaptación de documentos XML

XSL es el acrónimo de *Extensible Stylesheet Language*, que traduciríamos como "lenguajes de hojas de estilo extensibles".

El W3C creó tres especificaciones para el XSL:

1. **XSLT**: es un lenguaje XML diseñado para transformar documentos XML a otras sintaxis.
2. **XPath**: es un lenguaje de consulta que usamos para crear expresiones que recorren un documento XML a través de sus nodos para acceder a determinadas secciones.
3. **XSL-FO**: es un lenguaje XML que usamos para dar una presentación visual a un documento XML. Comúnmente usado para la generación de PDF.

5.1. Necesidades y ámbitos de aplicación

En 1997, grandes compañías informáticas, una de ellas Microsoft, pidieron crear un lenguaje de presentación y estandarización para documentos XML. A raíz de esa petición se formó el W3C Working Group, que desarrolló los lenguajes XSL.

Debemos tener en cuenta que diferentes documentos XML pueden tener DTD distintos, por lo que la estandarización en la presentación y la transformación de documentos XML a otras sintaxis se convierte en una necesidad.

Además, nos permite presentar la información XML de manera más legible para una impresión o una lectura por pantalla.

XSLT (*extensible Stylesheet Language for transformations*)

Es un lenguaje estándar de W3C que se recomienda por primera vez en 1999. Está basado en XML, por lo que, podemos decir que cada hoja de transformaciones es un documento XML bien formado.

Se trata de un lenguaje de programación (declarativo) que es capaz de generar documentos en diferentes formatos, como pueden ser: XML, HTML, texto, PDF, etc. partiendo de un documento XML.

Lenguaje declarativo: está basado en definir una serie de reglas o plantillas que deben ser aplicadas a un documento XML para conseguir transformarlo en la salida seleccionada.

Estas reglas, que suelen tener extensión *.xls*, unidas al documento *.xml* se van a pasar como parámetros a un procesador XSLT, que será el encargado de generar (como salida) el nuevo documento ya transformado.

Hojas de estilo: CSS vs XSLT

Existen dos familias diferentes de normas definidas por W3C: CSS y XSLT.

• **CSS:** ofrece la posibilidad de definir las propiedades asociadas a los diferentes elementos de marcado (letra negrita, de un tamaño determinado, con o sin bordes, etc.), aunque presenta una serie de inconvenientes:

- No puede cambiar el orden de los elementos pertenecientes a un documento HTML.
- No tiene permitido llevar a cabo distintas operaciones con los elementos.
- Los elementos no se pueden combinar entre ellos.

• **XSLT:** supera los inconvenientes anteriores, ya que puede determinar el contenido que tiene un documento XML y elegir el orden en el que se va a mostrar. También permite llevar a cabo las distintas operaciones que sean necesarias.

Es posible decantarse por una de las dos o incluso tenemos la opción de utilizar CSS y complementarlo con XSLT

Procesador XSLT

Formado por una serie de aplicaciones que permiten procesar aquellos documentos XML mediante las hojas de transformaciones XSLT.

- Primero, el procesador XSLT lee un documento XML.
- A continuación, puede representar el documento mediante un árbol de nodos. Estos nodos pueden ser elementos, atributos, texto, comentarios, instrucciones o espacios de nombres. Los nodos actúan de la misma forma que un árbol (padres, hijos, ascendientes, etc.).
- El procesador XSLT recorre y procesa nodo a nodo. El nodo que se está tratando en un instante determinado se denomina *nodo de contexto.*

Proceso de transformación

Para llevar a cabo este proceso de transformación debemos disponer del XML de partida junto con la hoja de estilos XSLT. Podemos llevarlo a cabo desde tres sitios diferentes:

- **En el servidor web**: utiliza una serie de lenguajes de servidor, como pueden ser: PHP, JSP, etc. para aplicar la XSLT al XML, de tal forma que permite enviar de vuelta el HTML.
- **En el cliente web**: en este caso, es el cliente el encargado de realizar las transformaciones necesarias mediante determinados programas, como JavaScript, que cuenta con una serie de funcionalidades para el tratamiento de documentos XML, transformaciones XSLT, etc.
- **En una aplicación diferente**: permite utilizar distintos programas independientes para llevar a cabo este proceso de transformación.

XML podemos llevar a cabo las transformaciones a:

- Texto
- XML
- HTML

Pasos para la transformación:

- Debe analizar la hoja de instrucciones, pasando a convertirse en una estructura árbol.
- Debe procesar el documento XML y convertirlo en una estructura árbol.
- El procesador XSLT debe posicionarse en la raíz del documento XML.
- Aquellos elementos que no formen parte del espacio de nombres (con prefijo *xsl*) se pasan a la cadena de salida sin que haya existido modificación alguna.
- El procesador XSLT solo puede aplicar una única regla a cada nodo.

5.2 Descripción y utilización de estructura, sintaxis y plantillas

Una hoja de transformaciones tiene que estar compuesta por lo siguiente:

- Declaración del documento XML:

```
<?xml version="1.0?">
```

- Elemento raíz:

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

- Espacio de nombres, donde xsl corresponde al prefijo:

```
http://www.w3.org/1999/XSL/Transform
```

- Otra serie de elementos que denominamos de nivel superior que es conveniente que aparezcan en la hoja de transformaciones, siempre como hijos de *<xsl:stylesheet>.*

Algunas transformaciones especiales

A continuación, vamos a ver una serie de casos en los que es conveniente hacer uso de diferentes trasformaciones especiales:

- **Sin plantillas**: El contenido de texto del XML se envía a la salida, ignorando atributos, comentarios e instrucciones de procesamiento.
- **Solo una plantilla**: Si solo hay una plantilla y está asociada al nodo raíz vacío, no se enviará nada a la salida.
- **Sin plantilla en el nodo raíz**: Se recorre el árbol del XML, aplicando reglas cuando se encuentra un elemento con una plantilla asociada.
- **Con plantillas**: Se recorre el XML en orden y, al encontrar un elemento con plantilla coincidente, se aplica la transformación.

Elementos básicos de una hoja de transformaciones

Existe un grupo de instrucciones que nos van a permitir generar un gran número de salidas. Destacaremos los siguientes:

- **xsl:stylesheet / xsl:transform**: Elementos raíz en la hoja de transformaciones.
    - **Atributos obligatorios**:
        - **versión**: Especifica la versión de XSLT.
        - **xmlns**: Define el espacio de nombres del documento XML.
    - **Atributos optativos principales**:
        - **exclude-result-prefixes**: Excluye ciertos espacios de nombres en la salida.
- **xsl:output**: Define el formato del documento de salida.
    - **Atributos principales**:
        - **method**: Tipo de salida (por defecto XML).
        - **versión**: Versión del formato de salida.
        - **encoding**: Juego de caracteres de salida.
        - **indent**: Controla la indentación.
        - **omit-xml-declaration**: Omite la declaración XML si es necesario.
        - **standalone**: Define si el documento es independiente.
- **xsl:template**: Define plantillas con reglas para transformar elementos XML.
- **xsl:apply-templates**: Aplica las plantillas definidas en el XML.

xsl:template

Aplica una plantilla a un nodo específico, reemplazándolo junto con sus descendientes por el resultado de la transformación. Este hecho nos va a eliminar toda la información de los descendientes.

Atributos principales:

- **name**: Asigna un nombre a la plantilla.
- **match**: Define el patrón XPath al que se aplicará la plantilla.
- **priority**: Controla la prioridad de la plantilla (valores entre -9 y 9).
- **mode**: Diferencia entre plantillas con el mismo match.

xsl:apply-templates

Es otra de las funciones principales en el ámbito del lenguaje de las transformaciones XSLT encargada de procesar y aplicar plantillas a los nodos seleccionados.

Atributos principales:

- **select**: Especifica los nodos que serán procesados.
- **mode**: Distingue entre plantillas con el mismo match.

Reglas de resolución de conflictos en plantillas

En XSLT, las plantillas tienen una prioridad asociada que determina cuál se aplica cuando múltiples plantillas coinciden con un mismo nodo. La prioridad puede variar de **-9 a 9**.
Si no se especifica una prioridad explícita mediante el atributo priority, se aplican las siguientes reglas automáticas:

Patrón	Prioridad
***, @*, text() (patrones genéricos)**	**-0.5**
prefijo:*, @prefijo:* (con espacio de nombres)	**-0.25**
producto, @unidades (elementos/atributos específicos)	**0**
proveedor/representante, producto/precio (jerarquías específicas)	**0.5**

Conflictos de Plantillas

Si varias plantillas tienen **la misma prioridad**, el comportamiento puede variar:

- **Error:** Dependiendo del procesador XSLT.
- **Última plantilla aplicada:** La última definida en el XSLT puede ser la aplicada.

Estas reglas ayudan a controlar cómo se procesan y transforman los documentos XML, garantizando que se aplique la plantilla más adecuada en cada caso.

Modos de las plantillas

Es bastante frecuente que una hoja de estilo acceda distintas veces a un mismo nodo, devolviendo cada vez valores diferentes a la salida. En estas ocasiones, debemos utilizar el atributo de las plantillas mode, que nos permite etiquetar cada plantilla para que se puedan llamar por el texto de la etiqueta.

xsl:call-template:

La podemos utilizar para llamar a una plantilla por su nombre. Cuando la aplicamos, no cambia el nodo de contexto, por tanto, en la plantilla que invoquemos tenemos que utilizar el mismo direccionamiento de aquellos elementos que existían en la que realizaba la llamada.

- **Atributos obligatorios:**
    - **name**: nombre de la plantilla a la que se va a llamar.

xsl:value-of:

Nos permite visualizar el contenido que se indica en el atributo *select* junto con todos sus descendientes.

- **Atributos obligatorios:**
    - **select**: expresión XPath que determina el elemento o atributo del que debemos extraer el contenido.
- **Atributos optativos principales:**
    - **Disable-output-escaping**: indica si algunos caracteres especiales (<,&) van a ser enviados a la salida tal cual o, por lo contrario, en su forma de entidad *(&lt, &amp).*

Versión simplificada para mostrar valores de los atributos

Solo es posible utilizar esta versión simplificada en determinados casos, como cuando se genera un nuevo elemento (etiqueta) en el documento de salida y deseamos asignar algún valor a alguno de sus atributos.

xsl:text: permite escribir un texto de forma literal en la salida.

- Atributos optativos principales:
    - **Disable-output-escaping**: indica si algunos caracteres especiales *(<,&)* van a ser enviados a la salida tal cual o, por lo contrario, en su forma de entidad *(&lt &amp).*

Instrucciones de control

Disponemos de un conjunto de etiquetas que ofrecen la posibilidad de reproducir el funcionamiento de aquellas instrucciones de control de flujo correspondientes a los lenguajes imperativos.

• **xsl:for-each**: permite repetir una lista de elementos para poder realizar diferentes transformaciones sobre ellos.

- Atributos obligatorios:
    - **Select**: expresión XPath que determina la lista de elementos que se van a repetir.
    - **xsl:sort**: devuelve los datos de un documento XML ordenados por algún método.
- Atributos optativos principales:
    - **select**: expresión XPath que determina el nodo o grupo de nodos que van a constituir el método de ordenación.
    - **lang**: permite seleccionar el idioma que vamos a emplear al ordenar.
    - **data-type**: tipo de datos que se van a ordenar.
    - **order**: indica si los elementos se van a ordenar de forma ascendente o descendente (ascending, descending).
    - **case-order**: determina el orden de letras mayúsculas (upper-first) o minúsculas (lower-first).

xsl:if: utilizado cuando existe comportamiento condicional. Realiza una pregunta por una condición y, si esa condición es cierta, va a realizar una cosa. En caso de que sea falso, realizará otra.

- Atributos obligatorios:
    - **test**: expresión XPath que, en caso de ser cierta, se puede aplicar a una plantilla.
    - **xsl:choose, xsl:when, xsl:otherwise**: ofrecen la posibilidad de llevar a cabo un comportamiento condicional con distintas opciones, además de otra por defecto.

5.3 Elaboración de documentación

Elaboración de un documento XML con hojas de estilo

Las hojas de estilo **CSS** se utilizan principalmente para dar formato a las páginas web en **HTML**, pero también pueden aplicarse a documentos **XML** para mejorar su presentación en navegadores compatibles.

En **junio de 1999**, el **W3C** aprobó una recomendación que establece cómo vincular documentos **XML** con hojas de estilo **CSS**. Para ello, se emplea la instrucción especial *<?xml-stylesheet?>,* que funciona de manera similar a la etiqueta *<link>* utilizada en **XHTML**.

Esta instrucción debe colocarse **al inicio del documento XML**, justo después de la declaración XML (*<?xml version="1.0"?>*), y en su atributo *href* se indica la ruta de la hoja de estilo, ya sea de manera **absoluta o relativa**.

Generación de nuevos elementos y atributos

Cuando se trabaja con **XSLT**, es posible decidir si la salida debe incluir elementos y atributos del documento original o si se deben generar nuevos. Para ello, XSLT ofrece diversas instrucciones que permiten crear, modificar o copiar elementos en el documento de salida.

1. Creación de Nuevos Elementos y Atributos

- ***xsl:element***: Permite generar un nuevo elemento en el documento de salida. Su atributo obligatorio *name* define el nombre del elemento, mientras que el atributo opcional *namespace* permite asignarle un espacio de nombres. También se puede utilizar *use-attribute-sets* para aplicar conjuntos de atributos.
- ***xsl:attribute***: Se usa para generar un nuevo atributo dentro de un elemento. Requiere el atributo *name* para definir su nombre y opcionalmente *namespace* para especificar el espacio de nombres correspondiente.

2. Comentarios e Instrucciones de Procesamiento

- ***xsl:comment***: Inserta un comentario en el documento de salida, que puede contener texto literal o valores extraídos del XML de origen.
- ***xsl:processing-instruction***: Permite incluir instrucciones de procesamiento en el documento de salida. Se usa, por ejemplo, para indicar hojas de estilo o configuraciones específicas.

3. Copia de Elementos

- ***xsl:copy-of****:* Genera una copia exacta de un elemento, incluyendo todos sus atributos y elementos descendientes.
- ***xsl:copy***: Copia solo un nodo específico sin incluir sus elementos internos. Se puede complementar con use-attribute-sets para agregar atributos al nodo copiado.

Conclusión

- XSLT proporciona herramientas versátiles para generar y modificar XML en la salida. Podemos:
- Crear nuevos elementos y atributos (xsl:element, xsl:attribute)
- Insertar comentarios e instrucciones de procesamiento (xsl:comment, xsl:processing-instruction)
- Copiar elementos del XML original (xsl:copy-of, xsl:copy)

5.4 XSL-FO

Como ya indicamos al comienzo de esta unidad formativa, **XSL-FO** (*XSL- Formatting Objects* u "objetos de formateo" *XSL*) es el lenguaje que se encarga de determinar el aspecto que van a tener los distintos datos a la hora de mostrarlos en un formato determinado de salida, que suele ser: PDF, RTF, PostScript, etc.

Procesadores XSL- FO

Hacen referencia a una serie de programas que recibe un documento XSL- FO de entrada para generar otro de salida con diferente formato.

FOP (*Formatting Objects Processor* - Procesador de Objetos de Formateo)

Este procesador es una aplicación (Java) dentro del código de Apache que, entre sus tareas principales, se encarga de coger un documento XSL-FO de entrada para poder generar una salida en un formato diferente, por ejemplo, PDF.

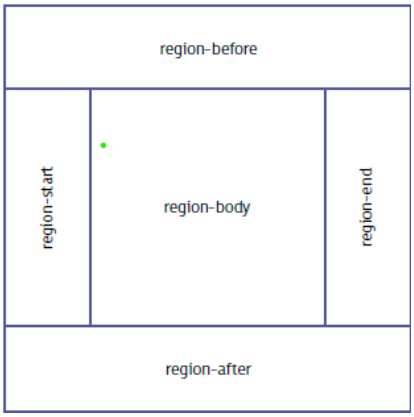
Objetos de formateo

Las etiquetas que se utilizan en este lenguaje están siempre relacionadas con distintos elementos de maquetado, como páginas, párrafos, bloques y tablas. En este caso, XSL-FO utiliza distintas áreas rectangulares para visualizar la salida. Entre sus áreas más importantes, destacamos:

Páginas: cuando la salida de una trasformación se organiza mediante páginas que, a su vez, están compuestas de regiones.

Regiones: contiene una serie de regiones, como:

- **region-body** (cuerpo de página).
- **region-before** (cabecera de la página).
- **region-after** (pie de página).
- **region-start** (lateral izquierdo).
- **region-end** (lateral derecho).
- **region-after** (pie de página).
- **region-start** (lateral izquierdo).
- **region-end** (lateral derecho).



Las diferentes regiones pueden contener áreas de bloque:

- **Áreas de bloque**: permiten definir elementos de bloque no muy grandes, como párrafos, líneas o incluso tablas. La mayoría de las áreas de bloque contienen áreas de línea.
- **Áreas de línea**: permiten definir un texto dentro de las áreas de bloque. Las áreas de línea pueden contener áreas secuenciales.
- **Áreas secuenciales**: permiten definir algún texto dentro de las áreas de líneas.

Estructura de una página XSL-FO

El elemento **<fo:layout-master-set>** contiene el patrón de la página o grupo de páginas **<fo:simple-master- page>** que dispone de patrones, márgenes, cabecera.

- **<fo:root>:** corresponde al elemento raíz.
  - **<fo:layout-master-set>:** patrones de las páginas del documento.
  - **<fo:page-sequence>:** páginas del documento.

Objetos de formateo para la estructura de documentos

1. fo:root

El elemento raíz de un documento XSL-FO, que contiene los siguientes objetos:

- **fo:layout-master-set**
- **fo:declarations** (opcional)
- **fo:page-sequence**

2. fo:layout-master-set

Este objeto contiene todos los modelos de página que se pueden utilizar en un documento XSL-FO, los cuales definen el diseño de las páginas.

3. fo:simple-page-master

Permite definir el formato (layout) de una página o un conjunto de páginas. Cada modelo de página puede tener hasta **cinco regiones**:

- **Cuerpo (region-body)**
- **Cabecera (region-before)**
- **Pie (region-after)**
- **Margen izquierdo (region-start)**
- **Margen derecho (region-end)**

De estas regiones, el **cuerpo** es la única obligatoria.

Propiedades principales:

- **master-name**: Define el nombre del modelo de página.
- **Estilos de página**: Como **márgenes**, **dimensiones de la página** (page-height, page-width), etc.

4. fo:declarations

Este objeto agrupa las declaraciones de estilo dentro de la hoja de formato. Actúa como un contenedor para los objetos de formateo que se utilizarán para generar la salida.

5. fo:page-sequence-master

Define el orden en el que se presentarán los objetos fo:simple-page-master en el documento.

Propiedades principales:

- **master-name**: Nombre del modelo que se aplicará.

6. fo:repeatable-page-master-alternatives

Este objeto permite manejar la repetición de un grupo de objetos.

fo:conditional-page-master-reference: Se emplea para definir las alternativas condicionales de la repetición de los modelos.

Propiedades principales:

- **maximum-repeats**: Establece el número máximo de veces que el modelo se puede repetir.

7. fo:conditional-page-master-reference

Solo se aplica si se cumplen las condiciones especificadas.

Propiedades principales:

- **master-reference**: Hace referencia al nombre del modelo fo:simple-page-master que se utilizará.
- **page-position**: Indica la posición ordinal de la página a la que se le aplicará el modelo.
- **odd-or-even**: Permite especificar si el modelo se aplicará a páginas **pares (even)** o **impares (odd)**.

Objetos de formateo para el contenido de las paginas

Utilizados para asignar un contenido determinado a cada página que forme parte del documento.

1. fo:page-sequence

Actúa como un contenedor para los objetos que forman las páginas del documento de salida. Dentro de un fo:page-sequence, se hace referencia a un objeto fo:page-sequence-master o fo:simple-page-master que define el formato y las características de las páginas.

Propiedades principales:

- **master-reference**: Indica el nombre del objeto fo:page-sequence-master o fo:simple-page-master al que se asociará la secuencia de páginas.

2. fo:title

Define el título de una secuencia de páginas. Este título es asignado al contenido de la secuencia de páginas y aparece generalmente en lugares como la cabecera del documento.

3. fo:static-content

Este objeto contiene elementos estáticos que se repiten en varias páginas, como cabeceras o pies de página. Su contenido no cambia entre las páginas y puede aparecer en múltiples lugares del documento.

Propiedades principales:

- **flow-name**: Define el área en la que se ubicará el contenido estático, como una cabecera o pie de página.

4. fo:flow

Este objeto contiene los elementos que se mostrarán en una página, tales como el texto o el contenido dinámico. Permite controlar el flujo de los elementos en la página.

Propiedades principales:

- **flow-name**: Define el área específica donde se posicionarán los elementos dentro del flujo de la página.

5. fo:block

Actúa como un contenedor a nivel de bloque, similar a un en HTML. Se utiliza para agrupar contenido y gestionar su disposición en la página, permitiendo crear estructuras jerárquicas y controladas.

6. fo:inline

Es un contenedor que organiza los elementos de forma secuencial, sin introducir saltos de línea, permitiendo distribuir el contenido en una línea continua dentro de la página.

Objeto de formateo para generar listas

En **XSL-FO**, se utilizan varios objetos de formateo para construir listas estructuradas. Estos elementos trabajan en conjunto para definir la apariencia y el contenido de las listas dentro de un documento.

1. fo:list-block

Este objeto se encarga de definir una lista en XSL-FO, actuando como contenedor de los elementos que la componen.

Contenido:

- **fo:list-item**: Representa cada elemento individual dentro de la lista.

2. fo:list-item

Cada ítem dentro de la lista se define con este objeto. Contiene dos partes esenciales: la etiqueta del ítem y el cuerpo del mismo.

Contenido:

- **fo:list-item-label**: Se encarga de mostrar la viñeta o marcador del ítem.
- **fo:list-item-body**: Contiene el texto o contenido principal del ítem.

2.1. fo:list-item-label

Define la etiqueta o marcador del ítem, que puede ser un número, un símbolo o cualquier otra indicación visual.

Contenido:

- Puede incluir elementos como:
    - **fo:block** (bloques de texto)
    - **fo:block-container** (contenedor de bloques)
    - **fo:table** (tablas)
    - **fo:list-block** (listas anidadas)
    - **fo:list-item** (otros elementos de lista)

2.2. fo:list-item-body

Este objeto contiene el contenido principal de un ítem de la lista, es decir, el texto o los elementos que lo describen.

Objetos de formateo para generar tablas

En **XSL-FO**, existen diversos objetos de formateo que permiten la creación y estructuración de tablas dentro de un documento. Estos elementos trabajan en conjunto para definir la apariencia y disposición de las tablas.

1. fo:table-and-caption

Este objeto actúa como un contenedor que agrupa los diferentes elementos de una tabla, incluyendo su título y estructura.

Contenido:

- **fo:table-caption** (título de la tabla, opcional)
- **fo:table** (estructura de la tabla)

2. fo:table-caption

Define el título o descripción de la tabla. Es un elemento descendiente de **fo:table-and-caption**.

3. fo:table

Se encarga de la definición de la tabla en sí, incluyendo sus filas, columnas y contenido.

4. fo:table-column

Permite configurar el formato de las columnas de la tabla.

Propiedad principal:

- **column-width**: Define el ancho de la columna.

5. fo:table-header

Hace referencia a la cabecera de la tabla, donde se incluyen los encabezados de cada columna.

Contenido:

- Puede contener filas (**fo:table-row**) o celdas (**fo:table-cell**).

6. fo:table-footer

Define el pie de la tabla, útil para mostrar información adicional en la parte inferior.

Contenido:

- Similar a la cabecera, puede contener filas (**fo:table-row**) o celdas (**fo:table-cell**).

7. fo:table-body

Es el contenedor principal de los datos de la tabla, agrupando las filas (**fo:table-row**) y sus respectivas celdas (**fo:table-cell**).

8. fo:table-row

Representa una fila dentro de la tabla.

Contenido:

- Contiene una o más celdas (**fo:table-cell**).

9. fo:table-cell

Define una celda dentro de la tabla, donde se muestra la información correspondiente a cada fila y columna.

Objetos de formateo para generar enlaces, imágenes

1. fo:basic-link:

Hace referencia a un enlace determinado del documento. Entre sus principales prioridades, destacamos:

- **internal-destination**: identificador a donde apunta el enlace al documento.
- **external-destination**: URI de la página a la que lleva el enlace.

2. fo:external-graphic:

Ofrece la posibilidad de insertar una imagen.

3. fo:leader:

Ofrece realizar una línea horizontal. Entre sus principales prioridades, destacamos:

- **leder-length**: hace referencia a la longitud de la línea.
- **leder-patten**: hace referencia al patrón de la línea.