

# ACTIVIDAD 3. XSLT Y XPATH

## LENGUAJE DE MARCAS



VERÓNICA BONIS MARTÍN

MARIA CARMEN CORREA HERAS

ÁNGEL SÁNCHEZ-SIERRA CRUZ

JOSÉ MARÍA TENREIRO EIRANOVA

**REPOSITORIO GITHUB**

[HTTPS://GITHUB.COM/ITDDAW/ACTIVIDAD3LDM.GIT](https://github.com/ITDDAW/ACTIVIDAD3LDM.GIT)

## Actividad 3. XSLT y XPATH

Dado el XML del enunciado de la actividad:

Realizar las siguientes requerimientos:

### Requerimiento 1

Mediante XSLT y XPATH, crear una página web en HTML y CSS en la que se plasme toda la información del XML. Dicha página web debe contener al menos los siguientes requisitos

1. Al menos dos tablas
2. Al menos dos enlaces
3. Al menos una lista ordenada o no ordenada
4. Al menos un formulario de contacto

A partir de estos requisitos, los alumnos podrán decidir hacer la página HTML a su gusto.

IMPORTANTE: Cada alumno propondrá una estructura de HTML resultante. Una vez todos los alumnos hayan puesto todas sus soluciones, se cotejará con el resto para poner la solución final donde se pondrán también los estilos CSS finales a la página.

Nota: Se valorarán los estilos utilizados (CSS)

Valoración: 7 puntos sobre 10

### Requerimiento 2

Mediante XSLT y XPATH, crear un nuevo documento XML el cual contenga la misma información, pero estructurada de forma diferente. Por ejemplo, poner atributos "id" como elementos, cambiar el nombre de las etiquetas, combinar valores de etiquetas en una sola, etc.

IMPORTANTE: Cada alumno propondrá una solución de XML resultante. Una vez todos los alumnos hayan puesto todas sus soluciones, se cotejará con el resto para poner la solución final, explicando el porque se tomó dicha decisión.

Valoración: 3 puntos sobre 10

### Detalles de entrega

Para toda la actividad se valorará el trabajo en equipo, el orden y la claridad de la documentación.

El código fuente estará alojado en un repositorio GITHUB.

Para la entrega, es necesario la creación de un pequeño documento formal sobre la actividad (portada, explicación, etc.), indicando los componentes del equipo, la URL del repositorio GITHUB compartido donde está el código fuente, las decisiones tomadas y la labor de cada integrante del equipo. También se valorarán la explicación de los problemas encontrados y su solución.

### CONSIDERACIONES PREVIAS

Para completar la tarea, cada persona trabaja por separado sus versiones de XSLT. Se hacen unas consideraciones previas:

- Trabajar en Visual Studio Code. Se recomienda instalar el plugin *XML Language Support by Red Hat* que nos facilitará la tarea de dar formato al archivo XML, corrección de errores, asignar sangrías automáticamente a los nodos, y que nos parece muy buena opción para trabajar. Además instalaremos el plugin *Live Server*, el cual nos servirá para la previsualización del trabajo a la hora de hacer cambios y comprobar el resultado final de la actividad.
- Probamos también otro editor específico: EDITIX *Professional XML Editor 2021 - EditiX*, que ofrece algunas ventajas y que tiene una versión gratuita bajo algunas condiciones
- Trabajar de nuevo con GIT, subiendo cada comentario y documentos a la carpeta compartida en el repositorio <https://github.com/ITDDAW/Actividad3LdM.git>

Una vez preparado el entorno de trabajo y el método, comenzamos a analizar el problema que nos plantea el ejercicio. Para ello nos parece interesante seguir este flujo:

- a) Crear una “estructura de árbol” en la que identifiquemos claramente la estructura jerárquica de los elementos que se incluirán en el XML (nodo raíz, subnodos, atributos...) que nos permita ir definiendo mejor las etiquetas.
- b) Definir el nodo raíz y los subnodos mediante etiquetas.
- c) Ir rellenando los atributos requeridos en cada elemento.

### FUNDAMENTACIÓN

XML no es un lenguaje sino un metalenguaje ya que permite crear lenguajes.

XML es un estándar desarrollado por el W3C [Estándares - W3C España](#), el cual desarrolla numerosas tecnologías que están basadas en XML.

XML proporciona una serie de reglas para que cualquiera pueda definir su propio conjunto de etiquetas y atributos, así como las relaciones entre las etiquetas. A esto se le llama la gramática del lenguaje.

En primer lugar, debemos preocuparnos de que el XML **esté bien formado**, esto es:

- Debe aparecer la declaración XML lo que se llama el prólogo:

`<?xml version = "1.0" encoding = "ISO-8859-1" standalone = "yes"?>`

- Si no se incluye los valores por defecto son: version 1.0, encoding UTF-8 y standalone yes.
- El cuerpo del documento se compone de elementos con etiqueta de apertura y de cierre. La etiqueta de inicio puede tener atributos con valor entrecomillado. Y por último entre las etiquetas de inicio y cierre el elemento puede tener contenido.
- *Solo puede existir un único elemento raíz, del que parten los demás.*
- Los elementos deben seguir una estructura jerárquica. Cada elemento debe estar contenido dentro de otro, exceptuando el elemento raíz.
- Los elementos deben *estar alineados*; deben contener tabulaciones de forma que en la columna que tenemos la etiqueta de inicio de un elemento, debe estar también la de cierre.
- Los *elementos no pueden estar superpuestos ni solapados*.
- Los *elementos deben estar cerrados*, es decir, todos los elementos deben contener su etiqueta de inicio y de cierre.
- El nombre de los elementos debe empezar siempre con un carácter no numérico y nunca puede comenzar por XML o cualquier variación, es decir, no puede empezar por xml, Xml, XML, xML, etc.
- Hay atributos reservados:
  - El prefijo xml se reserva para las especificaciones de xml
  - Xml : lang -----> para el idioma del elemento.
  - Xml : space : default | preserved -----> procesamiento normal o conservar los espacios en blanco.
  - Xml : id -----> se emplea como identificador único de todo el documento.
- Hay que respetar las mayúsculas y minúsculas. XML es un lenguaje case sensitive (sensible a mayúsculas y minúsculas).
- Existen caracteres especiales que se deben expresar con entidades específicas.
- Los valores de los atributos deben ir entre comillas dobles o simples. Si en el valor de un atributo se quieren escribir comillas dobles o simples se deben usar estas entidades que representan caracteres especiales:  
  
    &quot;            &apos;
- Se permiten elementos vacíos y se pueden escribir:

### Actividad 3. XSLT y XPATH

`<libro></libro>`      ó

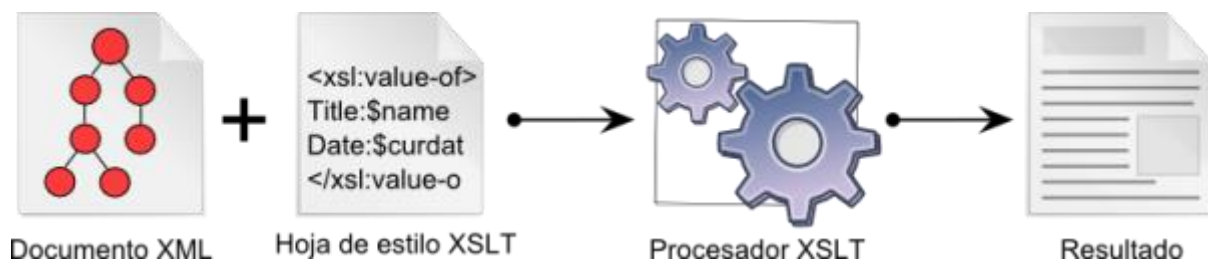
`<libro/>`

- Los comentarios se escriben `<!-- comentario -->`

**XSLT (eXtensible Stylesheet Language for Transformations)** es un lenguaje que permite aplicar una transformación a un documento XML para obtener otro documento XML, un documento HTML o un documento de texto plano.

La hoja de estilos XSLT con las reglas de transformación es también un documento de texto XML en sí, generalmente con extensión **.xsl**, por lo que se podrá comprobar si está **bien formado** o no.

El **funcionamiento** lo podemos observar en la siguiente imagen:



A un documento XML se le pueden aplicar una o varias transformaciones XSLT e incluso una transformación CSS. Las hojas de estilos XSLT son más útiles que las hojas de estilos CSS porque:

- Permiten cambiar el orden los elementos.
- Permiten realizar operaciones con sus valores.
- Permiten agrupar elementos.

De ahí que se suelen utilizar en combinación más que decantarse por una u otra hoja de estilos.

XSLT es un estándar del **W3C**:

- XSLT 1.0
- XSLT 2.0
- XSLT 3.0

El primer paso será definir en la cabecera del documento XML qué hoja de estilos XSLT se va a aplicar como vemos a continuación:

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

Recordamos que las hojas de estilos XSLT deben estar bien formadas, pero no van a validar.

### Actividad 3. XSLT y XPATH

La transformación se basa en el uso de plantillas (templates) con la etiqueta `<xsl:template>` que se aplicarán a cada nodo o atributo del documento según se va recorriendo éste, sustituyendo de esa manera el nodo por el contenido de su plantilla.

La hoja de estilos debe tener al menos una plantilla que suele corresponder con el nodo raíz, aunque puede ser de otro nodo. Si esta plantilla no contiene nada, como en el siguiente ejemplo, no se enviará nada como salida de la transformación:

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

En el interior de una plantilla podemos utilizar la etiqueta `<xsl:value-of>` para imprimir el valor que contiene el nodo, utilizando expresiones XPath para acceder a ellas.

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

Los atributos también pueden transformarse creándoles una plantilla:

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

Si utilizamos la etiqueta `<xsl:for-each>`, nos permite pasar por todos los nodos seleccionados, como si se tratara de un bucle "for" en programación:

```
<xsl:for-each select="bib">
  <ul>
    <xsl:for-each select="book">
      <li>
        <xsl:value-of select="title" />
      </li>
    </xsl:for-each>
```

### Actividad 3. XSLT y XPATH

**Requerimiento 1 :** Hemos realizado un diseño simple y claro de página web porque consideramos que lo importante en esta actividad es saber realizar la transformación de un archivo XML. Por lo tanto, después de aplicar la transformación al ejemplo XML dado por la actividad, nuestra página HTML quedaría de la siguiente manera:



The image shows a web page for 'Proeduca Instituto Tecnológico Edix'. It features two tables. The first table, titled 'CICLOS GRADO SUPERIOR', lists three cycles: ASIR (Administración de Sistemas Informáticos en Red) with decree 2009, DAW (Desarrollo de Aplicaciones Web) with decree 2010, and DAM (Desarrollo de Aplicaciones Multiplataforma) with decree 2010. The second table, titled 'PROFESORES', lists four professors: Félix, Raúl, Raquel, and Tomás, each with an ID from 1 to 4.

CICLOS GRADO SUPERIOR			
ID	NOMBRE	GRADO	DECRETO
ASIR	Administración de Sistemas Informáticos en Red	Superior	2009
DAW	Desarrollo de Aplicaciones Web	Superior	2010
DAM	Desarrollo de Aplicaciones Multiplataforma	Superior	2010

PROFESORES	
ID	NOMBRE
1	Félix
2	Raúl
3	Raquel
4	Tomás

Se han creado tablas con los elementos incluidos en los nodos “ciclos” y “profesores”. Para acceder a cada uno de los elementos (atributos ) definiendo las rutas teniendo en cuenta la estructura del XML.

Ejemplo: acceder a un atributo incluido dentro de otro elemento ( decretoTitulo)

```
<xsl:value-of select="decretoTitulo/@año" />
```

Para agilizar la creación de las tablas hemos utilizado la estructura repetitiva que se ejecutará para cada elemento dentro del nodo definido (en este caso ciclos).

```
<xsl:for-each select="ite/ciclos/ciclo">
```

### Actividad 3. XSLT y XPATH

Para los elementos incluidos dentro de ciclos y profesores hemos utilizado una estructura de tabla, que está a su vez incluida en una estructura <div> que nos permitirá dar un mejor formato a la web.

```
div class="ciclos">
  <table class="tabla2">
    <th colspan="4">CICLOS GRADO SUPERIOR</th>
    <tr>
      <th>
        ID ...
```

En la siguiente parte de la transformación utilizamos las listas (en este caso sin orden) dentro de otra estructura <div> que usamos para incluir dos listas y una imagen que contiene un enlace:

```
<div class="listaenlaces">
  <div class="administracion">
    <ul class="lista1">
      <h3>
        ADMINISTRACIÓN INSTITUTO
      </h3>
      <ul>
        <h4> DIRECCION</h4>
        <li>
          Nombre:...
```

Para usar el enlace que tenemos incluido en el XML usamos la siguiente estructura con XPath, accediendo al atributo dentro del elemento "ite".

```
<a>
  <xsl:attribute name="href">
    <xsl:value-of select="ite/@web" />
  </xsl:attribute>
  
</a>
```

Por último creamos un formulario completo en el que se han usado los elementos input\_type de texto para el nombre y apellidos ( con autofocus en el nombre):

```
<label for="nombre">Nombre Alumno: </label>
  <input type="text" name="nombre" id="nombre" autofocus="yes" />
  <br />
```

Elección de diferentes opciones dentro de una lista, con select – option , buscando de nuevo en la ruta requerida según la estructura del XML:



### Actividad 3. XSLT y XPATH

```
<label for="ciclo"> Ciclo matriculado </label>
  <br/>
  <select name="ciclo">
    <xsl:for-each select="ite/ciclos/ciclo">
      <option>
        <xsl:value-of select="nombre" />
      </option>
    </xsl:for-each>
  </select>
<br />
```

Se ha incluido también la opción de selección de sólo uno de los valores de una lista con “radio”:

```
<label for="puntuacion">Valoración</label>
  <br />
  <input type="radio" name="puntos" value="0" />
  Muy Negativa....
```

También se ha considerado la creación de una caja de texto multilínea para incluir comentarios, junto con las opciones de enviar ( submit) o borrar valores incluidos ( reset):

```
label for="comentarios">Deja a continuación tus impresiones</label>
  <br />
  <textarea name="comentarios" id="comenta" cols="50" rows="6"></textarea>
  <br />
  <input type="submit" value="Enviar encuesta" />

  <input type="reset" value="Limpiar encuesta" />
```

### Actividad 3. XSLT y XPATH

#### ADMINISTRACIÓN INSTITUTO

##### DIRECCION

Nombre: Chon

Despacho: Numero 31, 3ª Planta, Edificio A

##### JEFATURA DE ESTUDIOS

Nombre: Dani

Despacho: Numero 27, 2ª Planta, Edificio B



#### Encuesta de Satisfacción

Nombre Alumno:

Apellidos Alumno:

Ciclo matriculado

Administración de Sistemas Informáticos en Red ▼

Profesor a puntuar

Félix ▼

Valoración

- ☐ Muy Negativa  
☐ Negativa  
☒ Positiva  
☐ Muy Positiva

Enviar la encuesta a que persona:

Chon ▼

Deja a continuación tus  
impresiones

Enviar encuesta

Limpiar encuesta


### Actividad 3. XSLT y XPATH

**Requerimiento 2:** Para el requerimiento 2 se nos pide cambiar lo que consideremos en el XML y trasladar esos cambios a la transformación, hemos decidido incluir algunos elementos padres más, por ejemplo personal que podría englobar a la docencia y administración, hemos incluido links a los perfiles linkedin del personal y a páginas descriptivas para los diferentes ciclos (ASIR, DAM y DAW) y después hemos hecho cambios en la organización del nuevo html, cambiando alguna tabla por lista. Al final el resultado de los cambios sería el siguiente:

# Proeduca

## Instituto Tecnológico Edix

### CICLOS GRADO SUPERIOR



ID	NOMBRE	GRADO	DECRETO
ASIR	Administración de Sistemas Informáticos en Red	Superior	2009
DAW	Desarrollo de Aplicaciones Web	Superior	2010
DAM	Desarrollo de Aplicaciones Multiplataforma	Superior	2010

#### ADMINISTRACIÓN

**DIRECTOR**  
[Chon](#)  
Numero 31, 3ª Planta, Edificio A

**JEFE DE ESTUDIOS**  
[Dani](#)  
Numero 27, 2ª Planta, Edificio B

#### DOCENCIA

[Félix](#)  
Entornos de Desarrollo // Sistemas Informáticos

[Raúl](#)  
BBDD // Programación

[Raquel](#)  
Sistemas Informáticos

[Tomás](#)  
Programación // FCT

## Actividad 3. XSLT y XPATH

## Encuesta de Satisfacción

Nombre Alumno:

Apellidos Alumno:

Ciclo matriculado

Administración de Sistemas Informáticos en Red

Profesor a puntuar

Valoración

☐ Muy Negativa

☐ Negativa

☒ Positiva

☐ Muy Positiva

Enviar la encuesta a que persona:

Deja a continuación tus impresiones

Enviar encuesta

Limpiar encuesta

Tanto en el requerimiento 1 como en el requerimiento 2 la transformación a HTML la hemos hecho para que sea un modelo responsive ante cualquier dispositivo.

### **PASOS PARA TRABAJAR EN GRUPO**

La forma que ha tenido de trabajar el grupo ha sido mediante el repositorio GIT, en este repositorio se ha ido actualizando la página según cada miembro iba aportando algo a ella. Los pasos seguidos para ello en git han sido los siguientes:

- 1- Hay que decidir si se va a trabajar subiendo cambios con el perfil de Github del perfil creado (utilizar las contraseñas y nombre de usuario propias de ese perfil [ITTDAW@gmail.com](mailto:ITTDAW@gmail.com)) o utilizar nuestro propio perfil de Github y trabajar como colaborador (Se han enviado invitaciones a todos a los correos con los que cada persona se ha registrado en Github, hay que abrirlo y darle a aceptar)
- 2- Hacer un clone del repositorio en una carpeta de nuestro equipo  
Git clone <https://github.com/ITTDAW/Actividad2LdM.git>  
Entrar en la carpeta que se ha creado nueva dentro e iniciar el repositorio
- 3- Ahora estaremos en la rama master por defecto de nuestro repositorio local. Si se va a realizar cambios en el programa, se recomienda primero crear una rama propia para tener dos copias por si acaso.  
Git checkout -b nombrerama
- 4- Ahora ya estaremos en una nueva rama de nuestro repositorio local. Realizar los cambios que consideremos en el programa que estamos haciendo, todas las pruebas que consideremos, es el momento de probar todo.
- 5- Cuando pensamos que hemos hecho algo valido para subir al repositorio, vamos a añadir nuestros cambios en nuestro repositorio local y rama propia.  
Git add .  
Git commit -m "Los cambios que hemos hecho"
- 6- Ahora ya tenemos nuestros cambios en nuestro repositorio local y rama propia, podríamos verlos con git log --oneline. Para subir estos cambios al repositorio remoto, es necesario que nos cambiemos a la rama master y fusionar nuestra rama personal con la master.  
Git checkout master  
Git merge nombredenuestrarama
- 7- En estos momentos nuestra rama master ya puede subirse al repositorio remoto. Si lo hacemos desde nuestro perfil de github, hemos aceptado el correo de aceptación de colaboradores y en nuestro equipo ya tenemos las credenciales de nuestro perfil de github, solamente con git push origin master se deberían de subir los cambios. Si trabajamos desde el perfil de ITTDAW tendremos que loguearnos cuando lo pida el gitbash y no tener guardadas nuestras credenciales en Windows de nuestro perfil personal.
- 8- CONSEJOS: Siempre antes de empezar a trabajar, hacer git pull en la rama master de tu repositorio local para ver si algún compañero ha hecho algún cambio, después fusionar esos cambios con tu rama personal y entonces si empezar a trabajar.
- 9- Si a alguien se le complica su repositorio local, que no se preocupe, se hace otro git clone en su equipo y vuelve a empezar desde el punto 2.