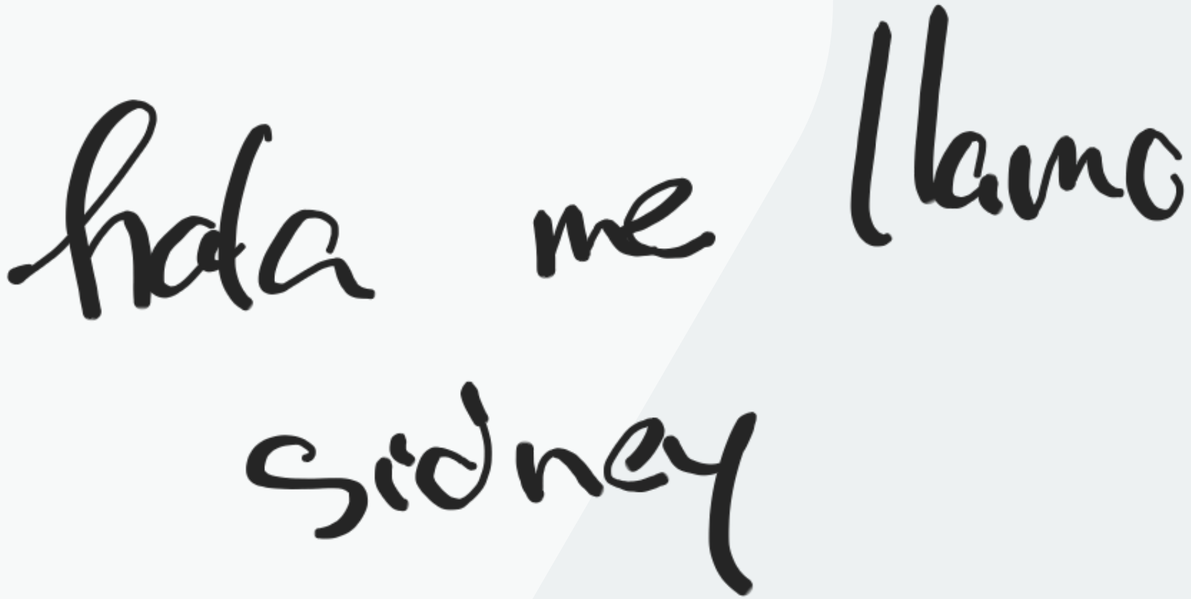
LMGSI 22/23

XSLT



Unidad 8

Rafa Domínguez

Necesidad de transformar

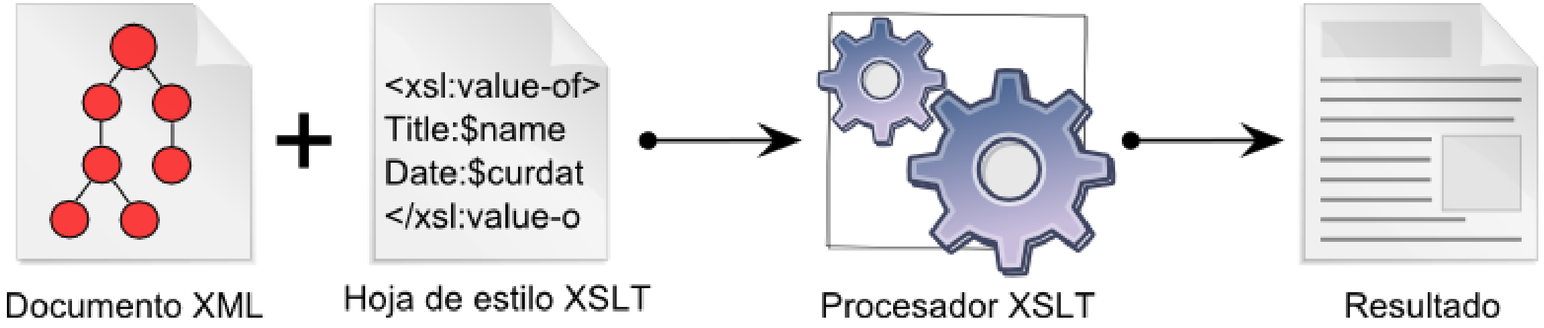
- XML es un estándar para “transmitir” datos a través de Internet asociado a una reglas de validación (XSD y DTD) para compatibilizar su uso.
- Necesitamos transformar los XML en medios de procesamiento y/o reproducción de información funcionales como HTML, texto, etc.
- XSL(eXtensible Stylesheet Language) es una familia de tecnologías de W3C que permiten transformar documentos XML.
- Lo podemos transformar en cualquier cosa que se pueda representar en **cadenas de carácteres**: HTML, CSV, sentencias SQL, codigo de programa, imagenes vectoriales, etc.

Familia XSL

- **XSLT(XSL Transformation):** es el lenguaje de transformación propiamente dicho.
- **XSL-FO(XSL Formatting Objects):** es el lenguaje en el que se indica el formato que debe tener el xml para representarse en diferentes dispositivos o medios como pantallas o papel.
- **XPath(XML Path Languaje):** es un lenguaje que permite recorrer un xml utilizando expresiones. Se utiliza en XSLT para seleccionar partes de los documentos.

XSLT(XSL Transformation):

- Es un lenguaje **declarativo**, ya que consiste en la declaración de una serie de **reglas o plantillas** que hay que aplicar a un documento xml para transformarlo en el output deseado.
- Una regla asocia un patrón (**expresión**) con elementos del documento xml de partida y les aplica una serie de acciones.
- Estas reglas se almacenan en un documento de texto, normalmente .xsl que, junto al xml serán pasados como parámetros a un procesador XSLT, que generará el documento transformado.



XSLT vs CSS

- CSS no puede:
  - Cambiar el orden en que los elemntes de un HTML se visualizan (filtrar u ordenar por algún criterio.
  - Realizar operaciones como sumar los valores de elemento por ejemplo.
  - Combinar múltiples elementos. Por ejemplo combinar todas las nóminas de un empleado para saber el total anual de ingresos y retenciones.
- Se pueden **combinar**, con XSLT se puede determinar que contenido mostrar y en que orden, realizar calculo sencillos, etc. y con CSS dar un formato agradable de presentación.

Procesador XSLT I

- Son apps que procesan xml mediante hojas de transformaciones xslt.
- Cuando el procesador lee el doc xml genera una representación de dicho documento en forma de **árbol de nodos.**
- Los tipos de nodo son: elemento, atributo, texto, comentario, instrucción de procesamiento y espacio de nombres.
- Con relaciones de tipo árbol genealógico.
- Los **comentarios y las instrucciones de procesamiento** son partes del árbol y deben ser **tenido en cuenta** a la hora de contar los nodos o iterar sobre ellos.
- El procesadr va recorriendo y procesando **nodo a nodo.**
- El nodo que se está tratando en un momento dado se denomina **nodo de contexto**.

Procesador XSLT II

- Procesadores online:
  - xsl.online-toolz
  - shell-tools
  - markbucyan.appspot
- Instalables:
  - Navegadores web
  - Xalan de Apache
- Hay editores que permien realizar las transformaciones:
  - Altova StyleVision
  - <oXygen/>
  - XML Copy Editor

Proceso de transformación

- Para realizar una transformación se precisa un xml y un xslt.
- Una transformación puede realizarse en:
  - Un servidor web: mediante un lenguaje de servidor (php, .net, etc.) aplica el xsl al xml y envia de vuelta un html al cliente.
  - Un cliente web: el cliente realiza las transformaciones. Por ejemplo, Javascript junto a las librerias Sarissa permiten hacerlo.
  - Mediante una aplicación independiente. Hay programas que permiten realizar transformaciones.

Pasos para la transformación

1. El xslt es analizado y se convierte en una estructura de árbol.
2. El xml es procesado y convertido en una estructura de árbol.
3. El procesador xslt se posiciona en la **raíz** del xml.
4. Los elementos (etiquetas html por ejemplo) que **no** formen parte del **espacio de nombres** (prefijo xsl) son pasados a la **cadena de salida** sin modificarse. Estos se denominan elementos de **resultado literal.**
5. El procesador xslt **sólo aplica una regla a cada nodo**. Si tenemos dos reglas para el mismo, solo aplica la **última en aparecer**.

Salvo en el caso anterior, el orden en el que aparecen las plantillas en el xslt no es representativo. Lo que marca el orden es el recorrido del árbol por el procesador.

Nota técnica importante

Las políticas de seguridad de los navegadores pueden bloquear las trasformaciones xslt al abrir lo xml. Para evitarlo podemos:

- Alojar el xml y el xsl en un servidor web (local o remoto) y acceder al xml mediante http.
- Modificar la configuración de seguridad del navegador.

En firefox la configuración se modifica de la forma siguiente:

1. Abrir navegador.
2. Escribir en la barra de direcciones **about:config**
3. Pulsar sobre aceptar riesgo y continuar.
4. Buscar parámetro **privacy.file_unique_origin** y cambiar el valor a **false**.

Etructura básica de un xslt

Se compone por:

1. Una **declaración** de documento xml.
2. Un **elemento raíz (xsl:stylesheet)**.
3. El espacio de nombres (http://www.w3.org/1999/XSL/Transform) del prefijo **xsl.**
4. Normalmente habrá el elemento **<xsl:template match="/"></xsl:template>** que indica que se debe aplicar la transformación a todo el documento xml.
5. Existen otros elementos, llamados de **nivel superior**, que de aparecer, siempre lo harán como hijos xsl:stylesheet. (xsl:import, xsl:output, xsl:template, xsl:key, etc.)

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

Conceptos básicos transformación

- En xslt se alternan texto con elementos xsl i los textos se volcarán en la salida sin sufrir ninguna modificación.
- El ejemplo siguiente aplicado a un xml generará una salida consistente en el código html incluido en **<xsl:template>** y coge los valores de los elementos xml mediante **<xsl:value-of select="elemento/subelemento"/>**

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="ejemplo1.xslt"?>
<pelicula>
    <titulo> Superman: la película</titulo>
    <argumento>Un extraterrestre con increíbles poderes se dedica a salvar a los habitates del planeta tierra</argumento>
</pelicula>
```

Ejemplo 1 con html

```xml
<?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 lang="en">
            <head>
                <meta charset="UTF-8"/>
            </head>
            <body>
                <h2>
                    <xsl:value-of select="pelicula/titulo"/>
                </h2>
                <p>
                    <xsl:value-of select="pelicula/argumento"/>
                </p>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Resultado ejemplo 1 con html

Superman: la película

Un extraterrestre con increíbles poderes se dedica a salvar a los habitates del planeta tierra

Elementos XSLT I

- Hay muchos elementos para crear transformaciones. Algunos de los más relevantes son:
- **<xsl:value-of select="XPath">:** obtiene el valor de un elemento o atributo.
- **<xsl:for-each select="xPath">:** recorre un conjunto de elementos.
- **<xsl:sort select="elemento a ordenar">**: orden aun conjunto de elementos.
- **<xsl:if test="expresión">:** permite condicionar la transformación en función de una condición.

Elementos XSLT I

- **<xsl:choose>**
  - **<xsl:when test="expresión">...<xsl:when>**
  - **<xsl:otherwhise>...<xsl:otherwhise>**
- **</xsl:choose> :** utiliza conjuntamente **when y otherwise** para condicionar en función de múltiples opciones.
- **<xsl:template match="ruta_plantilla">:** permite establecer una plantilla y determinar sobre qué conjunto de elementos se va a realizar la transformación.
- **<xsl:output paràmetros>:** define el formato de salida del documento.
- **<xsl:comment>**comentario que no será procesado **</xsl:comment>**

Ejercicio conjunto (Objetivo)

- El objetivo es crear la siguiente tabla a partir de xml dado.

País	Año	Título	Autor	Museo
Francia	1976	Bal au moulin	Pierre-Auguste Renoir	Museo de Orsay
Francia	1979	Pont de Maincy	Paul Cézanne	Museo de Orsay
Países Bajos	1890	Trigal con cuervos	Vincent Willm van Gogh	Museo de Van Gogh

Ejercicio conjunto (XML)

- Hay que analizar que partes componen el xml y la información que contiene

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="ejemplo2.xsl"?>
<obras>
    <obra pais="Francia" anyo="1976">
        <titulo>Bal au moulin</titulo>
        <autor>Pierre-Auguste Renoir</autor>
        <museo visible="true">Museo de Orsay</museo>
    </obra>
    <obra pais="Francia" anyo="1979">
        <titulo>Pont de Maincy</titulo>
        <autor>Paul Cézanne</autor>
        <museo visible="false">Museo de Orsay</museo>
    </obra>
    <obra pais="España" anyo="1822">
        <titulo>Amalia de Llano y Dotres</titulo>
        <autor>Federico de Madrazo y Kunt</autor>
        <museo visible="true">Mueseo del prado</museo>
    </obra>
    <obra pais="España" anyo="1823">
        <titulo>Saturno devorando a su hijo</titulo>
        <autor>Francisco José Goya y Luciente</autor>
        <museo visible="true">Museo del Padre</museo>
    </obra>
    <obra pais="España" anyo="1635">
        <titulo>El príncipe Baltasar Carlos a caballo</titulo>
        <autor>Diego Rodríguez de Silva y Velázquez</autor>
        <museo visible="false">Museo del Prado</museo>
    </obra>
    <obra pais="Países Bajos" anyo="1890">
        <titulo>Trigal con cuervos</titulo>
        <autor>Vincent Willm van Gogh</autor>
        <museo visible="true">Museo de Van Gogh</museo>
    </obra>
</obras>
```

Ejercicio conjunto (Análisis)

- El resultado es una tabla html, por tanto necesitaremos una **plantilla** html para ejecutarla: **<xsl:template match="/">**
- También tiene colores por lo que tendremos que poner **css**, podemos usar la etiqueta **style** de html.
- El xml contiene un elemento **raíz** llamado **obras**, que en su interior contiene **más** de un elemento **obra**.
- Cada elemento obra tiene **2 atributos**, que entenderemos como fijos, que proporciona información sobre el elemento: **pais** y **anyo**.
- Obras tiene 3 elementos hijos: **titulo, autor y museo**.
- Museo tiene un **atributo boolean** llamado **visible,** que podremos utilizar para visualizar o no ese elemento.

Ejercicio conjunto (Desarrollo xslt I)

- El xml es la versión 1.0 por lo que debe declararse en el xsl para evita incompatibilidades.

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

- También debemos definir el espacio de nombres que utilizaremos durante el trabajo.

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

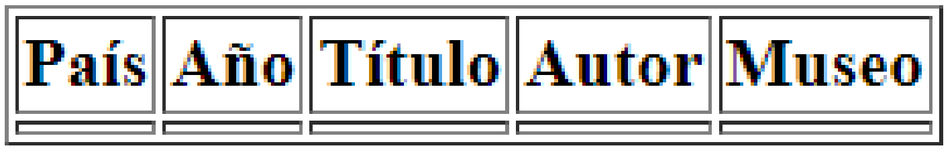
- Así como indicar que usaremos una plantilla y que estará en este propio documento.

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

Ejercicio conjunto (Desarrollo xslt II)

- Hay que definir la estructura html base y la tabla

```
<?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>
        <style>
        </style>
        <body>
            <table border="1">
                <tr>
                    <th>País</th>
                    <th>Año</th>
                    <th>Título</th>
                    <th>Autor</th>
                    <th>Museo</th>
                </tr>
                <tr>
                    <td></td>
                    <td></td>
                    <td></td>
                    <td></td>
                    <td></td>
                </tr>
            </table>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```



Ejercicio conjunto (Desarrollo xslt III)

- Para obtener la información de los elementos xml **obra** y sus atributos, recorreremos el xsl con un for-each.
- Durante el recorrido ordenaremos la información por **titulo** con un **sort.**
- Si nos fijamos, la obra más antigua que aparece es de 1890, por tanto debemos crear un filtro de años con un **If**.
- Para recoger el contenido de los atributos utilizaremos **@nombre_atributo.**

```
<table border="1">
    <tr>
        <th>País</th>
        <th>Año</th>
        <th>Título</th>
        <th>Autor</th>
        <th>Museo</th>
    </tr>
    <xsl:for-each select="obras/obra">
        <xsl:sort select="titulo"/>
        <xsl:if test="@anyo &gt; 1890">
            <tr>
                <td></td>
                <td></td>
                <td></td>
                <td></td>
                <td></td>
            </tr>
        </xsl:if>
    </xsl:for-each>
</table>
```

Ejercicio conjunto (Desarrollo xslt IV)

- Ahora debemos recuperar el contenido del xml que queremos que aparezca en cada celda de nuestra tabla, para ello utilizaremos **value-of** (tanto para elementos como para atributos).

```
<xsl:for-each select="obras/obra">
    <xsl:sort select="titulo"/>
    <xsl:if test="@anyo &gt; 1889">
        <tr>
            <td><xsl:value-of select="@pais"/></td>
            <td><xsl:value-of select="@anyo"/></td>
            <td><xsl:value-of select="titulo"/></td>
            <td><xsl:value-of select="autor"/></td>
            <td><xsl:value-of select="museo"/></td>
        </tr>
    </xsl:if>
</xsl:for-each>
```

Ejercicio conjunto (Desarrollo xslt V)

- Ya disponemos de nuestra estructura de transformación completa.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/T
<xsl:template match="/">
    <html>
        <style>
        </style>
        <body>
            <table border="1">
                <tr>
                    <th>Pais</th>
                    <th>Año</th>
                    <th>Titulo</th>
                    <th>Autor</th>
                    <th>Museo</th>
                </tr>
                <xsl:for-each select="obras/obra">
                    <xsl:sort select="titulo"/>
                    <xsl:if test="@anyo &gt; 1889">
                        <tr>
                            <td><xsl:value-of select="@pais"/></td>
                            <td><xsl:value-of select="@anyo"/></td>
                            <td><xsl:value-of select="titulo"/></td>
                            <td><xsl:value-of select="autor"/></td>
                            <td><xsl:value-of select="museo"/></td>
                        </tr>
                    </xsl:if>
                </xsl:for-each>
            </table>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

País	Año	Título	Autor	Museo
Francia	1976	Bal au moulin	Pierre-Auguste Renoir	Museo de Orsay
Francia	1979	Pont de Maincy	Paul Cézanne	Museo de Orsay
Países Bajos	1890	Trigal con cuervos	Vincent Willm van Gogh	Museo de Van Gogh

Ejercicio conjunto (Desarrollo xslt VI)

- Le ponemos el código CSS para dejarlo bonito

```
<style>
th{
        background-color:black;
        color:white;
        padding:0.5em;
    }
    td{
        padding:0.5em;
    }
    tr:nth-child(even){
        background-color:green;
    }
    tr:nth-child(odd){
        background-color:yellow;
    }
</style>
```

País	Año	Título	Autor	Museo
Francia	1976	Bal au moulin	Pierre-Auguste Renoir	Museo de Orsay
Francia	1979	Pont de Maincy	Paul Cézanne	Museo de Orsay
Países Bajos	1890	Trigal con cuervos	Vincent Willm van Gogh	Museo de Van Gogh

Ejercicio conjunto (Desarrollo xslt VI)

- usar el atributo visible de la etiqueta museo

```
<xsl:for-each select="obras/obra">
    <xsl:sort select="titulo"/>
    <xsl:if test="museo/@visible = 'true'">
    <!--<xsl:if test="@anyo &gt; 1889">-->
        <tr>
            <td><xsl:value-of select="@pais"/></td>
            <td><xsl:value-of select="@anyo"/></td>
            <td><xsl:value-of select="titulo"/></td>
            <td><xsl:value-of select="autor"/></td>
            <td><xsl:value-of select="museo"/></td>
        </tr>
    </xsl:if>
</xsl:for-each>
```

País	Año	Título	Autor	Museo
España	1822	Amalia de Llano y Dotres	Federico de Madrazo y Kunt	Mueseo del prado
Francia	1976	Bal au moulin	Pierre-Auguste Renoir	Museo de Orsay
España	1823	Saturno devorando a su hijo	Francisco José Goya y Luciente	Museo del Padre
Países Bajos	1890	Trigal con cuervos	Vincent Willm van Gogh	Museo de Van Gogh