Tema 5: Transformació de documents XML

Índex de continguts

5.0 Introducció a la UF2.

5.1 Introducció a XSLT.

En capítols anteriors hem vist que XML és un format relativament llegible, doncs es tracta de fitxers de texte pla i la informació està organitzada jeràrquicament. No obstant, com hem vist darrerament amb HTML, als humans els agrada visualitzar la informació col.locada en determinats formats que facin la lectura més agradable, com l'al.lineació, format del texte o els colors.

Donat que XML està pensat principalment per emmagatzemar i intercanviar informació, si volem representar-les d'una altra manera, tindrem diverses possibilitats:

- Desenvolupar un programa que agafi les dades XML i generi la sortida en el format que volem. En aquest cas tenim l'inconvenient que hem de tenir coneixements de programació.
- Utilizar CSS, que com hem vist permet canviar la visualització de les dades d'un document HTML a la web. CSS permet canviar la visualització de les dades, però no transformar els documents com veurem en aquesta unitat.
- Transformar el document. Aquesta opció permet transformar el document en un altre pensat per a la seva visualització: PDF, HTML, etc.

5.2 eXtensible Stylesheet Language (XSL)

XSL és una familia de llenguatges basats en XML que permeten transformar els documents XML en altres documents. Consta del següents llenguatges:

- XSLT (XSL transformations) és un llenguatge per transformar documents XML.

- XPath és un llenguatge que permet accedir a parts concretes d'un document XML mitjançant expressions de búsqueda. Permet navegar per un document XML.
- XSL-FO (XSL formatting objects) permet definir format als documents XML, tant per a formats de pantalla com per a formats paginats. Permet la creació de documents PDF.

En aquest capitol ens centrarem bàsicament en la transformació de documents mitjantçant XSLT, que permet crear, a partir d'un document XML altres documents com XML, HTML o texte.

5.3 Fulls d'estil: CSS vs XSLT

L'organisme W3C ha definit dues families de normes per als fulls d'estil. La més sencilla es CSS, que ja hem vist al capítol dedicat a HTML. CSS té les següents limitacions, que es solucionen amb XSLT:

- Amb CSS no podem canviar l'ordre dels elements que surten en un document HTML, no es poden ordenar els elements o filtrar per algun criteri (tal com fariem a la clàusula WHERE de SQL).
- Amb CSS no podem realitzar operacions amb els elements, per exemple calcular els valors de tots els elements <preu> d'un document XML.

També podem combinar els dos llenguatges, CSS per donar estil al document i XSLT per fer transformacions sobre ell. A continuació tenim un exemple de XML i CSS.

Exercici: copia els següents codis en dos fitxers de texte i intenta mostrar-los en un navegador.

Document alumnes.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="estilo.css"?>
    <alumne>
        <nom>Marc</nom>
        <cognom>Màrquez</cognom>
    </alumne>
    <alumne>
        <nom>Fernando</nom>
        <cognom>Alonso</cognom>
    </alumne>
    <alumne>
        <nom>Pau</nom>
        <cognom>Gasol</cognom>
    </alumne>
```

Document estilo.css

```
alumne {
    text-align:center;
}
nom {
    font-weight:bold;
    font-family:tahoma;
    color:blue;
}
cognom {
    color:cyan;
}
```

5.4 Processadors XSLT

Un processador XSLT permet llegir un document XSL i a partir de les regles de transformació sobre un document XML, permet generar un altre document formatejat tipus XML, HTML o texte.

El seu ús implica necessàriament els següents passos:

1. Tenir un document XML ben definit (sintàcticament).
2. Validar el document mitjançant DTD o XML Schema (XSD).
3. Crear un full d'estil XSL ben format (ha de seguir les regles XML).
4. Vincular el document XML amb el full d'estil XSL.
5. Executar la tranformació amb el processador XSLT.

El processador XSLT va llegint el document XML, processant node a node el XML i aplicant les transformacions necessàries definidas en las **templates rules** del full XSLT. Los templates son un conjunto de reglas que se aplican cuando se encuentra alguno de los nodos especificados.

Podem treballar amb processadors off-line amb editors com **XML Copy Editor**, **Visual Studio Code** (aquest requereix un processador extern apart de una extensió) processadors en línia com:

- XSL Transformation (XSLT) Online Toolz
- XSLT Saxon Processor <– requereix Java

5.4.1. Crear full d'estil ben format

La declaració d'un document XSL és la següent (són equivalents):

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

o

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

5.4.2 Vincular un XSL al document XML

Per vincular el document XSL al document XML, afegim la següent declaració al XML:

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

5.4.3 Aplicar una plantilla al document XSL

Dintre del document XSL afegirem un template per indicar sobre quins nodes del document XML volem actuar ("/" indica l'arrel del document).

```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>
            <body>
                <h2>La meva col.lecció de CDs</h2>
                <!-- Aqui podem seleccionar elements XML i inserir-los al nostre HTML-->
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

5.5 Elements de plantilles més comuns.

Dintre de les nostres plantilles XSL podem utilitzar les següents instruccions de plantilla. Pertanyen al vocabulari XSLT definit a l'espai de noms XSL.

Els més utilitzats són:

- **xsl:value-of**. Inserta el valor d'un element o atribut XML a la sortida resultant. Podem fer servir l'atribut select per seleccionar l'atribut o subelement el valor del qual s'utilitzarà.

Exercici: copia els següents codis en dos fitxers de texte i fes la transformació amb el processador on-line. Respon a la següent pregunta: Quants discs aparèixen al fitxer html?

```xml
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
    <cd>
        <title>Empire Burlesque</title>
```

```
        <artist>Bob Dylan</artist>
        <company>Columbia</company>
        <price>10.90</price>
        <year>1985</year>
    </cd>
    <cd>
        <title>Hide your heart</title>
        <artist>Bonnie Tyler</artist>
        <company>CBS Records</company>
        <price>9.90</price>
        <year>1988</year>
    </cd>
</catalog>
```

```
<?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>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <tr>
        <td><xsl:value-of select="catalog/cd/title"/></td>
        <td><xsl:value-of select="catalog/cd/artist"/></td>
      </tr>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

- **xsl:for-each**. Es fa servir per a recórrer els elements d'un document i realitzar una sèrie d'operacions amb els nodes. L'atribut select determina els elements que s'ha de recórrer. És equivalent a l'instrucció foreach dels llenguatges de programació.

La seva sintaxi es:

```
<xsl:for-each select="condicio">
    <xsl:value-of select="..."/>
</xsl:for-each>
```

Exercici: modifiqueu l'exercici anterior per mostrar tots els albums:

```
<xsl:for-each select="catalog/cd">
```

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

- **xsl:if**. Podem sel.leccionar la informació que mostrarà per pantalla en funció de les condicions que li indiquem.

Exemple

```
<xsl:if test="price&lt;=10 and Artist='Xavier Camunyes'">
```

Compte amb les cometes simples si el que volem és comparar cadenes. En la següent taula es resumeixen els símbols utilitzats per als operadors booleans:

Símbol	Operador
=	Igual
!=	Distint
<	menor que
>	major que
< =	menor o igual que
> =	major o igual que
and	I lògic (&&)
or	O lògic (||)

- **xsl:sort**. Ens permet ordenar la informació en funció del contingut d'un element. Té més opcions que la clàusula ORDER BY a SQL.

La seva sintaxi seria:

```
<xsl:sort select="condició"/>
```

Compte: xsl:sort no té etiqueta de tancament

Els atributs (opcions) que té es mostren a la següent taula:

Atribut	Valor	Significat
select	XPath Expression	Especifica quin node o conjunt de nodes ordenar
lang	language-code	Especifica quin llenguatge s'utilitza per a l'ordenació
datatype	text, number, qname	Especifica el tipus de dada que s'ordenarà. Per defecte és "texte"
order	ascending, descending	Especifica el tipus d'ordre. Per defecte es "ascending"

Atribut	Valor	Significat
caseorder	upper-first, lower-first	Especifica si les lletres majúscules o minúscules s'ordenen en primer lloc

Exemple:

```
<xsl:for-each select="catalog/cd">
    <xsl:sort select="artist" order="descending"/>
    <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
    </tr>
</xsl:for-each>
```

- **xsl:choose**. Element per a condicionar els resultats que permet establir múltiples condicions dintre del recorregut de l'arbre XML. És equivalent a l'instrucció **switch-case-default** dels llenguatges de programacio.

La seva sintaxi seria:

```
<xsl:choose>
    <xsl:when test="expressio_1">
        tractament de les dades
    </xsl:when>
    <xsl:when test="expressio_2">
        tractament de les dades
    </xsl:when>
    ...
    <xsl:otherwise>
        tractament de les dades
    </xsl:otherwise>
</xsl:choose>
```

Exemple:

```
<xsl:for-each select="catalog/cd">
    <tr>
        <td><xsl:value-of select="title"/></td>

        <xsl:choose>
            <xsl:when test="price &amps;gt; 10">
                <td bgcolor="#ff00ff"><xsl:value-of select="artist"/></td>
            </xsl:when>

            <xsl:otherwise test="price &amps;gt; 10">
                <td><xsl:value-of select="artist"/></td>
```

```
            </xsl:otherwise>
        </xsl:choose>
    </tr>
</xsl:for-each>
```

- **xsl:output**. L'element output defineix el format de sortida del document XSL (XML, HTML o texte). És un element de nivell superior, per tant ha d'anar just sota l'element xsl:stylesheet o xsl:transform. Per exemple, el següent codi produeix un document XML com a sortida.

```
<?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" version="1.0" encoding="UTF-8" indent="yes"/>
    ...
    ...
</xsl:stylesheet>
```

- **xsl:include**. L'element include també és un element de nivell superior que inclou els continguts d'un full d'estils dintre d'un altre. Sintaxi:

```
<xsl:include href="URI">
```

5.6 Transformació de XML a XML (xsl:output)

5.7 Plantilles XSL.

Fins ara hem aplicat la plantilla a tot el document, i per crear els diferents estils als elements feiem servir xsl:if o xsl:choose. Ara farem servir diferentes plantilles per a condicionar el contingut de cada element.

Exemple

```
<?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>
                <body>
                    <h1>My CD Collection</h1>
                    <xsl:apply-templates/>
                </body>
            </html>
        </xsl:template>

        <xsl:template match="catalog">
            <h2>CD Catalog</h2>
            <table>
                <tr bgcolor="#887788">
                <th>Title</th><th>Artist</th>
                </tr>
```

```
            <xsl:apply-templates select="cd"/>
        </table>
    </xsl:template>

    <xsl:template match="cd">
        <tr>
            <xsl:apply-templates select="title"/>
            <xsl:apply-templates select="artist"/>
        </tr>
    </xsl:template>

    <xsl:template match="title">
        <td bgcolor="DDEEDD">
            <xsl:value-of select="."/>
        </td>
    </xsl:template>

    <xsl:tempalte match="artist">
        <td bgcolor="AABBAA">
            <xsl:value-of select="."/>
        </td>
    </xsl:template>
</xsl:stylesheet>
```