UF2. NF2. CONVERSIÓ DE DOCUMENTS XML: XSLT (EXTENSIBLE STYLESHEET LANGUAGE TRANSFORMATIONS)

Índex

1.INTRODUCCIÓ....2
1.1.Què és XSLT?....2
1.2.Per què es transformen els documents XML?....2
2.ELEMENTS SIMPLES....4
2.1.L'element <xsl:template>....5
2.2.L'element <xsl:apply-templates>....5
2.3.L'element <xsl:value-of>....6
2.4.L'element <xsl:copy>....6
2.5.L'element <xsl:copy-of>....8
3.ESTRUCTURES DE CONTROL....9
3.1.L'element <xsl:if>....9
3.2.L'element <xsl:choose>....10
3.3.L'element <xsl:for-each>....11
3.4.L'element <xsl:sort>....12
4.Modes XSL....13
5.Paràmetres i variables XSLT....15
6.L'element <xsl:output> 16

1. INTRODUCCIÓ

XSLT (extensible Stylesheet Language Transformations) és un estàndard de W3C. És un llenguatge funcional dissenyat principalment per transformar documents XML en altres estructures XML i no XML.

W3C va començar a desenvolupar el XSL (eXtensible Stylesheet Language, expressió anglesa traduïble com "llenguatge extensible de fulles d'estil") com una família de llenguatges basats en l'estàndard XML que permet descriure com la informació continguda en un document XML qualsevol ha de ser transformada o formatada per a la seva presentació en un mitjà.

Aquesta família està formada per tres llenguatges:

- **XSLT** (sigles d'Extensible Stylesheet Language Transformations, llenguatge de fulles extensibles de transformació), que permet convertir documents XML d'una sintaxi a altra (per exemple, d'un XML a un altre o a un document HTML).
- **XSL-FO** (llenguatge de fulles extensibles de format d'objectes), que permet especificar el format visual amb el qual es vol presentar un document XML, és usat principalment per a generar documents PDF.
- **XPath sintaxis** (no basada en XML) per a accedir o referir-se a porcions d'un document XML.

1.1. Què és XSLT?

XSLT és la part més important de XSL. S'usa per transformar un document XML en altre tipus de document, que serà reconegut per un navegador, com HTML i XHTML.

Amb XSLT podem afegir o eliminar elements i atributs del fitxer de sortida XML. Podem a més ordenar elements i prendre decisions sobre que elements poden ser o no visualitzats.

XSLT usa XPath per buscar la informació en un document XML, és a dir per navegar al llarg del document.

El resultat de la transformació pot ser:

- El mateix document XML (mateixes etiquetes), però amb filtres de sortida.
- Altre document XML (diferents etiquetes)
- Altres documents, que poden ser XML o no:
    - XHTML
    - Text
    - WAP

1.2. Per què es transformen els documents XML?

Principalment per dos motius:

- Per adaptar-los a altre model de dades. XML es defineix per a intercanviar dades entre aplicacions. Cada aplicació pot tenir diferents models de dades (és a dir, diferents tipus d'etiquetes).
- Per fer-los més llegibles: XML és massa complicat per llegir. Necessitem una transformació prèvia.

Altra cosa que hem de tenir en compte és que un document XSLT és un document XML, així que com a tal ha de complir la sintaxi de XML.

Un document XSL barreja just dues famílies d'etiquetes; aquelles que són de XSL i aquelles del marcat del document de sortida.

Exemple: Vegem com transformar un document XML en un document XHTML.

El document XML és:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<cataleg>
      <cd>
            <titol>Empire Burlesque</titol>
            <artista>Bob Dylan</artista>
            <pais>USA</pais>
            <companyia>Columbia</companyia>
            <preu>10.90</preu>
            <any>1985</any>
      </cd>
      .
      .
</cataleg>
```

Creem un document XSL amb la regla de transformació:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
   <body>
      <h2>La meva colecció de CDs</h2>
      <table border="1">
            <tr bgcolor="#9acd32">
                  <th>Títol</th>
                  <th>Artista</th>
            </tr>
            <xsl:for-each select="cataleg/cd">
            <tr>
                  <td><xsl:value-of select="titol"/></td>
                  <td><xsl:value-of select="artista"/></td>
            </tr>
            </xsl:for-each>
      </table>
   </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Vegem una mica del document XSL, en el mateix tenim:

- Els documents XSL tenen **només un element <xsl:stylesheet>**
- **Les regles de plantilla es defineixen com elements <xsl:template>**
- Les plantilles de sortida són continguts d'elements <xsl:template> i defineixen la transformació.

Finalment, per enllaçar el document XML al document XSL hem d'afegir:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="cdcatalogo.xsl"?>
<cataleg>
      <cd>
            <titol>Empire Burlesque</titol>
            <artista>Bob Dylan</artista>
            <pais>USA</pais>
            <companyia>Columbia</companyia>
            <preu>10.90</preu>
            <any>1985</any>
      </cd>
      .
      .
</cataleg>
```

2. ELEMENTS SIMPLES

Per als exemples següents anem a utilitzar el següent document XML, que us poso a continuació:

```
<persones>
   <persona>
      <nom>Albert Einstein</nom>
       <descripcio>
      Albert Einstein va ser un dels científics més famosos del segle XX. A ell,
      entre altres coses, li devem la Teoria de la Relativitat.
      </descripcio>
   </persona>
   <persona>
      <nom>Marie Curie</nom>
      <descripcio>
      Pionera en el camp de la radioactivitat. Va ser la primera persona es
      rebre dos premis Nobel i la primera dona en ser professora en la
      Universitat de París.
      </descripcio>
   </persona>
   <persona>
      <nom>Nikola Tesla</nom>
      <descripcio>
      Va ser inventor, enginyer mecànic, enginyer elèctric i un dels promotors
      més importants del naixement de l'electricitat comercial.
      </descripcio>
   </persona>
</persones>
```

El document persones.xslt associat és:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <xsl:template match="/">
      <html>
            <head>
                  <title>Informació sobre
                  <xsl:value-of select="count(/persones/persona)" />
                  científics</title>
            </head>
            <body>
                  <h3>Informació sobre
                  <xsl:value-of select="count(/persones/persona)" />
                  científics </h3>
                  <br/>
                  <xsl:apply-templates select="/persones/persona" />
            </body>
      </html>
   </xsl:template>
   <xsl:template match="persona">
      <h3><xsl:value-of select="nom" /></h3>
      <p><xsl:value-of select="descripcio" /></p>
      <br />
   </xsl:template>
</xsl:stylesheet>
```

2.1. L'element <xsl:template>

Usem l'element `<xsl:template match="...">` per seleccionar diferents parts del document XML. Per exemple, per seleccionar l'element arrel:

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

Així, si el processador de XSLT troba un node en l'arbre que sigui un node arrel, l'estructura corresponent al contingut d'aquest node és afegit a l'arbre resultat.

2.2. L'element <xsl:apply-templates>

L'etiqueta <xsl:apply-templates> és usada per a aplicar una plantilla (template) sobre l'element actual o sobre algun dels seus nodes fills.

Una vegada que es troba l'etiqueta <xsl:apply-templates> es buscarà alguna de les templates definides en el document XSL que coincideixi amb l'expressió XPath continguda en el seu atribut match.

Si afegim un atribut select a l'etiqueta <xsl:apply-templates> aquesta processarà solament el node o nodes fills que coincideixin amb el valor de l'atribut. També podem usar l'atribut select per a especificar l'ordre en el qual els nodes fills seran processats.

En l'exemple anterior cada vegada que el processador XSLT trobi un node persona que correspongui amb la trajectòria /persones/persona, el contingut de la plantilla és processada i el

contingut és afegit a l'arbre de resultats.

2.3. L'element <xsl:value-of>

Normalment el que volem és extreure informació de l'arbre del document. XSLT posseïx diverses formes d'usar la informació continguda en l'arbre.

L'element <xsl:value-of> proporciona el valor per una banda de l'arbre. Té un atribut select, el valor del qual és una expressió XPath.

2.4. L'element <xsl:copy>

L'element <xsl:copy> copia un node a l'arbre resultat, però no copia els nodes descendents. Això és útil si volem canviar l'estructura d'un element o el seu contingut, o afegir o eliminar atributs.

Vegem un exemple:

```
<persones>
   <persona>
      <nom>Lluís</nom>
      <cognom>Pérez</cognom>
   </persona>
   <persona>
      <nom>Anna</nom>
      <cognom>Rojo</cognom>
   </persona>
   <persona>
      <nom>Gerard</nom>
      <cognom>Puig</cognom>
   </persona>
</persones>
```

Per a l'exemple anterior, al que anomenarem "Persones2.xml", li aplicarem el següent full d'estils xslt, al qual anomenarem "Persones2.xsl":

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0" >
<xsl:template match="/">
   <persones>
      <xsl:apply-templates select="/persones/persona" />
   </persones>
</xsl:template>
<xsl:template match="persona">
   <xsl:copy>
      <xsl:attribute name="nom">
            <xsl:value-of select="nom"/>
      </xsl:attribute>
      <xsl:attribute name="cognom">
            <xsl:value-of select="cognom"/>
      </xsl:attribute>
   </xsl:copy>
</xsl:template>
</xsl:stylesheet>
```

Després de provar l'exemple, anem a analitzar-lo. Com anteriorment, existeix un template que té l'atribut matches="/". En comptes de crear elements literals HTML/XHTML, s'afegix l'element literal <persona>. L'element <xsl:apply-templates> és usat amb la ruta XPath /persones/persona.

Existeix un template que té match a "persona", el que coincideix amb el valor de l'atribut "select" de l'element "xsl:apply-templates". Per tant, per a cada node "persona" en el document font, el template especifica com processar-lo.

L'element xsl:copy és usat quan el node de context és un node element persona (en el segon template). Per tant, cada node element "persona" és afegit a l'arbre resultat, però sense copiar els nodes fill.

En aquest punt, el que tindríem si el document acabés en el primer element xsl:copy tindríem el següent resultat:

```
<persones>
        <persona/>
        <persona/>
        <persona/>
<persones>
```

El template empra l'element xsl:attribute per a afegir un nou node atribut a l'element persona en l'arbre resultat. En la línia <xsl:attribute name="nom"> estem dient que l'atribut es anomenarà "nom". I amb la línia <xsl:value-of select="nom"/> vam indicar que el valor de l'atribut serà el contingut de l'element "nom".

El mateix ocorre amb l'atribut "cognom".

2.4.1. Afegir elements fills

Si necessitem seguir el camí invers al de l'exemple anterior, podem fer-lo amb en el següent exemple. Partim del resultat generat en l'exemple anterior.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0" >
   <xsl:template match="/">
      <persones>
            <xsl:apply-templates select="/persones/persona" />
      </persones>
   </xsl:template>
   <xsl:template match="persona">
      <xsl:copy>
            <xsl:element name="nom">
                  <xsl:value-of select="@nom"/>
            </xsl:element>
            <xsl:element name="cognom">
                  <xsl:value-of select="@cognom"/>
            </xsl:element>
      </xsl:copy>
   </xsl:template>
</xsl:stylesheet>
```

Si observem el codi, es torna a repetir més o menys el mateix que en l'anterior. Tenim un template amb match a "/".

També tenim un element "xsl:apply-templates" amb select a "/persones/persona". Per a aquest "xsl:apply-templates" hi ha un template (l'atribut del qual match val "persona") que indica com processar cada node element "persona".

Amb el primer <xsl:copy>, vam copiar a l'arbre resultat el node element "persona". Deixem obert l'element <xsl:copy>, perquè dintre col·locarem altres elements.

Dintre de l'element <xsl:copy> vam crear nous elements, amb l'element <xsl:element>. A cada

element li assignem un nom. Per exemple amb ":element name="nom">" hem indicat que s'ha de crear un un nou node element anomenat nom.

A la seva vegada, dintre d'aquests nous elements, vam col·locar elements de tipus xsl:value-of que recuperen el contingut de cada atribut. Per exemple, la línia <xsl:value-of select="@cognom"/> recupera el valor de l'atribut "cognom".

2.5. L'element <xsl:copy-of>

Aquest element realitza una còpia en profunditat. És a dir, copia un node, juntament amb tots els seus nodes atribut i nodes descendents.

```
<enviament>
   <origen>Palencia</origen>
   <desti>Bilbao</desti>
   <adreça>
      <carrer>Laments,5</carrer>
      <ciudad>Gotham</ciutat>
      <pais>ES</pais>
      <cp>54321</cp>
   </adreça>
</enviament>
```

El full d'estils associat és:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0" >
   <xsl:template match="/">
      <notificacio>
            <xsl:apply-templates select="/enviament/desti" />
            <xsl:apply-templates select = "/enviament/origen" />
            <xsl:apply-templates select="/enviament/adreça" />
      </notificacio>
   </xsl:template>
   <xsl:template match="desti">
      <xsl:element name="origen">
            <xsl:value-of select="." />
      </xsl:element>
   </xsl:template>
   <xsl:template match="origen">
      <xsl:element name="desti">
            <xsl:value-of select="." />
      </xsl:element>
   </xsl:template>
   <xsl:template match="adreça">
      <xsl:copy-of select="." />
   </xsl:template>
</xsl:stylesheet>
```

Com podem veure, hem utilitzat l'element "copy-of" dintre d'un "template" aplicat al node "adreça". És a dir, hem copiat el node adreça, així com a tots les seves descendents. D'aquesta manera podem integrar tota aquesta zona de l'arbre en l'arbre resultat.

3. ESTRUCTURES DE CONTROL

Fins a ara hem vist que les transformacions es realitzen en un sol sentit, conforme es va instanciant cada template. No obstant això, podem crear diferents templates i aplicar un o altres en funció d'alguna condició.

3.1. L'element <xsl:if>

L'element if comprova si una certa condició booleana és veritable o falsa. Així, una instrucció condicional pot determinar si una acció es porta a terme o no. La seva sintaxi és:

```
<xsl:if test=condicion>
      …
</xsl:if>
```

Vegem un exemple, tenim un document amb informació sobre la memòria RAM de certs equips de la nostra empresa i volem generar a partir d'ell altre document on s'informi de certs ordinadors massa antics:

```
<ordinadors>
      <equip memoria=”4096”>B302PC01</equip>
      <equip memoria=”512”>B302PC02</equip>
      <equip memoria=”2048”>B302PC03</equip>
      <equip memoria=”1024”>B302PC04</equip>
</ordinadors>
```

El document XSLT que hauríem d'utilitzar és el següent:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <xsl:template match="/">
      <html>
            <head>
                  <title>Equips obsolets</title>
            </head>
            <body>
                  <h2>La memòria és massa baixa</h2>
                  <xsl:apply-templates select="/ordinadors/equip" />
            </body>
      </html>
      </xsl:template>
      <xsl:template match="equip">
            <xsl:if test="@memoria&lt;1024">
                  <p><strong><xsl:value-of select="." /></strong> té poca
                  memòria</p>
```

```
            </xsl:if>
      </xsl:template>
</xsl:stylesheet>
```

3.2. L'element <xsl:choose>

Amb aquest element podem triar entre vàries opcions excluyents entre sí. Podem indicar més possibilitats d'elecció. La sintaxis d'aquest element és la següent:

```
      <xsl:choose>
            <xsl:when test=condicio1>
            …
            </xsl:when>
            <xsl:when test=condicio2>
            …
            </xsl:when>
            …
            <xsl:otherwise>
            …
            </xsl:otherwise>
      </xsl:choose>
```

Utilitzem choose en l'exemple anterior:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <xsl:template match="/">
      <html>
            <head>
                  <title>Equips </title>
            </head>
            <body>
                  <h2>RESUM DELS EQUIPS</h2>
                  <xsl:apply-templates select="/ordinadors/equip" />
            </body>
      </html>
   </xsl:template>
   <xsl:template match="equipo">
      <xsl:choose>
            <xsl:when test="@memoria&lt;1024">
                  <p><strong><xsl:value-of select="." /></strong> té poca
                  memoria</p>
            </xsl:when>
            <xsl:when test="@memoria=1024">
                  <p><strong><xsl:value-of select="." /></strong> té memòria
                  normal</p>
```

```
            </xsl:when>
            <xsl:otherwise>
                  <p><strong><xsl:value-of select="." /></strong>té molta
                  memòria</p>
            </xsl:otherwise>
      </xsl:choose>
      </xsl:template>
</xsl:stylesheet>
```

3.3. L'element <xsl:for-each>

Per poder mostrar tots els nodes en un “nodo-set” utilitzem l'element for-each. Aquest element permet que tots els nodes puguin ser processats d'acord a les instruccions que hi ha dins d'un element “for-each”.

Vegem un exemple:

```
<?xml version="1.0" encoding="UTF-8"?>
<ordinadors>
      <equip memoria=”4096”>
            <caracteristica tipus=”id”>B302PC01</caracteristica>
            <caracteristica tipus=”marca”>Genèric</caracteristica>
      </equip>
      <equip memoria=”512”>
            <caracteristica tipus=”id”>B302PC02</caracteristica>
            <caracteristica tipus=”marca”>Genèric</caracteristica>
      </equip>
      <equip memoria=”2048”>
            <caracteristica tipo=”id”>B302PC03</caracteristica>
            <caracteristica tipo=”marca”>HP</caracteristica>
      </equip>
      <equip memoria=”1024”>
            <caracteristica tipus=”id”>B302PC04</caracteristica>
            <caracteristica tipus=”marca”>HP</caracteristica>
      </equip>
</ordinadors>
```

Podríem aplicar el següent full d'estils:

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

```
            </head>
            <body>
                  <h2>RESUM DELS EQUIPS</h2>
                  <xs:apply-templates select=”/ordinadors/equip” />
            </body>
      </html>
   </xsl:template>
   <xsl:template match=”equip”>
      <ul>
            <xsl:for-each select=”caracteristica”>
            <li><xsl:value-of select=”.”/></li>
            </xsl:for-each>
      </ul>
   </xsl:template>
</xsl:stylesheet>
```

3.4. L'element <xsl:sort>

Aquest element s'empra per mostrar la sortida dels elements ordenada. Es pot utilitzar amb combinació amb “apply-templates” i “for-each”. Per a l'exemple anterior, podríem aplicar el següent full d'estils:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <xsl:template match="/">
      <html>
            <head>
                  <title>Equips</title>
            </head>
            <body>
                  <h2>RESUM DELS EQUIPS</h2>
                  <xsl:apply-templates select=”/ordinadors/equip” >
                        <xsl:sort select=”@memoria”/>
                  </xsl:apply-templates>
            </body>
      </html>
   </xsl:template>
   <xsl:template match=”equip”>
      <h3><xsl:value-of select=”@memoria”/></h3>
      <ul>
            <xsl:for-each select=”caracteristica”>
                  <xsl:sort select=”.” order=”descending” />
```

```
                <li><xsl:value-of select=”.”/></li>
            </xsl:for-each>
      </ul>
   </xsl:template>
</xsl:stylesheet>
```

Si provem aquest exemple, podrem observar que l'ordre de la memòria s'ha establert per ordre alfabètic, quan el normal hagués estat per ordre numèric. Per a això, hem d'usar l'atribut “data-type/”, de la següent manera:

```
            <xsl:sort select=”@memoria” data-type=”number” />
```

Aquest atribut pot tenir els següents valors:

- `text`, és el valor per defecte
- `number`
- `qname`, es refereix a tipus definits per l'usuari.

4. Modes XSL

Fins a ara, quan hem utilitzat un element apply-templates, ho hem relacionat amb un template, per a això, fèiem correspondre l'atribut select de apply-templates amb l'atribut match de template. No hem tingut ambigüitats, perquè per a cada apply-templates només havia un template. Però, si hem d'utilitzar diversos template amb el mateix valor per a match, com els distingirem?

Vegem un exemple on hauríem d'utilitzar l'anterior, quan hàgim de processar un mateix node diverses vegades en el document.

```
<?xml version="1.0" encoding="UTF-8"?>
<article>
   <autors>
      <autor>Mauricio Aznar</autor>
      <autor>Esperanza Rojo</autor>
      <autor>José María Soler</autor>
   </autors>
   <titol> Fonaments de XML </titol>
   <capitols>
      <capitol numero="1" titol="Introducció"> Introducció a XML</capitol>
      <capitol numero="2" titol="XPath"> Llenguatge XPath</capitol>
      <capitol numero="3" titol="XSLT"> L'ús XSLT</capitol>
      <capitol numero="4" títol="XQuery"> Connexió amb les bases de dades
      </capitol>
   </capitols>
</article>
```

Suposem que volem generar a partir del document xml un document html que tingui el següent aspecte:

```
<html>
   <head>
      <title>Fonaments de XML</title>
```

```
   </head>
   <body>
      <h3>Fonaments de XML</h3>
      <p>Autors: Mauricio Aznar, Esperanza Rojo i José María Soler.</p>
      <h2>Índex</h2>
      <p><strong>1:</strong>Introducció</p>
      <p><strong>2:</strong>XPath</p>
      <p><strong>3:</strong>XSLT</p>
      <p><strong>4:</strong>XQuery</p>
      <h3>1.Introducció</h3>
      <p>Introducció a XML</p>
      <h3>2.XPath</h3>
      <p>Llenguatge XPath</p>
      <h3>3.XLST</h3>
      <p>L'ús de XSLT</p>
      <h3>4.XQuery</h3>
      <p> Connexió amb les bases de dades</p>
   </body>
</html>
```

Com es pot observar, el títol dels capítols s'empra dues vegades: la primera vegada per a la taula de continguts, i la segona durant el desenvolupament del text. Això vol dir, que el document XSLT que necessitarem utilitza dos template amb l'atribut match amb valor “capitol”.

Com els distingirem entre si quan els referenciem des de l'element “apply-templates” ?

La solució és la següent:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
   <html>
      <head>
            <title><xsl:value-of select="/article/titol"/></title>
      </head>
      <body>
            <h3><xsl:value-of select="/article/titol"/></h3>
            <p>Autors: <xsl:apply-templates select="/article/autors/autor"/></p>
            <h2>Índex</h2>
            <xsl:apply-templates select="/article/capitols/capitol"
            mode="Índex"/>
            <xsl:apply-templates select="/article/capitols/capitol"
            mode="Desenvolupament"/>
      </body>
   </html>
```

```
</xsl:template>
<xsl:template match="autor">
   <xsl:choose>
      <xsl:when test="position() < last()-1"> <xsl:value-of select="."/>
      ,</xsl:when>
      <xsl:when test="position() = last()-1"> <xsl:value-of select="."/> i
      </xsl:when>
      <xsl:otherwise> <xsl:value-of select="."/>.</xsl:otherwise>
   </xsl:choose>
</xsl:template>
<xsl:template match="capitol" mode="Índex">
   <p>
      <strong><xsl:value-of select="@número"/>:</strong>
      <xsl:value-of select="@títol"/>
   </p>
</xsl:template>
<xsl:template match="capitol" mode="Desenvolupament">
   <h3><xsl:value-of select="@número" />. <xsl:value-of select="@títol"/></h3>
   <p><xsl:value-of select="."/></p>
</xsl:template>
</xsl:stylesheet>
```

Observa que s'utilitza l'atribut mode, que distingeix un template d'un altre.

5. Paràmetres i variables XSLT

XSLT permet especificar variables i paràmetres mitjançant els elements xsl:variable i xsl:parameter. Ambdós poden ser referenciats usant $nomVariable o $nomParametre.

Nota: No cal confondre les variables en XSLT amb les variables en altres llenguatges, ja que una vegada que el seu valor és fixat no es pot modificar.

Vegem un exemple de la seva utilització: Suposar que tenim un document XML amb els noms i edats d'una sèrie de persones i volem que a l'introduir el nom ens aparegui l'edat.

```
<?xml version="1.0" encoding="UTF-8"?>
<edats>
   <persona nom=”Anna” edat=”21” />
   <persona nom=”Montse” edat=”19” />
   <persona nom=”Antonio” edat=”41” />
   <persona nom=”Lluís” edat=”35” />
</edats>
```

El full d'estils XSLT que haurem d'aplicar és el següent:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:parameter name=”persona” />
<xsl:template match="/">
   <html>
      <head>
            <title>Trobar l'edat usant un paràmetre XSLT</title>
      </head>
      <body>
            <xsl:apply-templates select=”/edats/persona[@nom=$persona]” />
      </body>
   </html>
</xsl:template>
<xsl:template match=”persona”>
   <p>L'edat <xsl:value-of select=”$persona” /> és <xsl:value-of
   select=”@edat” /> </p>
</xsl:template>
</xsl:stylesheet>
```

Per passar el paràmetre des de la línia de comandes usem la següent sintaxi:

```
java -jar saxon.jar -o edats.html edats.xml edats.xslt persona=”Anna”
```

6. L'element <xsl:output>

Amb XSLT es poden generar documents XML, HTML o de text. Podem triar de quin tipus serà el resultat mitjançant l'element **<xsl:output method=”tipus de document a generar” />**, on tipus de document a generar pot ser xml, html o text.