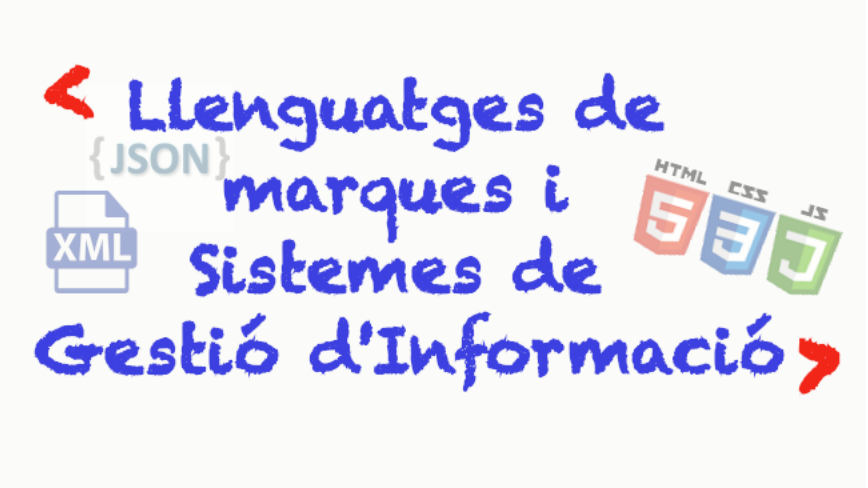
UNITAT 6. CONVERSIÓ I ADAPTACIÓ DE DOCUMENTS PER A L'INTERCANVI D'INFORMACIÓ



IES Sant Vicent Ferrer
Algemesí

Contingut

1. Introducció

A pesar que l'XML és un format relativament llegible en ser visualitzat, aquest no és un dels seus objectius principals, i menys si es té en compte que als humans els agrada llegir les dades col·locades en determinats formats que els facen la lectura més agradable.

En determinats casos pot caldre transformar els documents XML perquè siguen més fàcils de visualitzar, o també adaptar-los perquè puguen ser llegits per programes específics.

XML està pensat sobretot per a **emmagatzemar** i **intercanviar informació**, de manera que si cal representar les dades d'una manera diferent per optimitzar un procés o per millorar-ne la visualització hi haurà diverses possibilitats:

- **Desenvolupar un programa**: com que és relativament senzill treballar amb XML, es podria desenvolupar un programa que agafe les dades XML i genere la sortida tal com la volem. Això té l'inconvenient que caldrà tenir coneixements de programació i que pot representar molta feina encara que el que calga fer siga trivial.
- **Fer servir CSS**: en molts casos una solució senzilla seria fer servir CSS per representar la informació de manera més amigable fent servir un navegador. Però només serveix per canviar la visualització, no per canviar el document.
- **Transformar el document**: una solució alternativa consisteix a transformar el document en un altre que estiga pensat per ser visualitzat. Hi ha molt formats que estan pensats sobretot per ser visualitzats: PDF, HTML, XHTML, etc.

1.1. Ús de CSS

En molts casos si l'objectiu és que el document siga visualitzat d'una manera més agradable la solució més senzilla pot ser visualitzar-lo mitjançant CSS.

CSS és interessant sobretot quan es vulguen fer adaptacions senzilles. Si es té un document XML com el següent, i es vol representar com una targeta de vista, es pot fer servir un document CSS que continga les línies de l'exemple:

```xml
<?xml version="1.0" ?>
<professor>
    <nom>Marcel</nom>
    <cognom>Garcia</cognom>
    <departament>Departament d'Informàtica</departament>
    <carrecs>
       <carrec>Cap de Departament</carrec>
       <carrec>Tutor</carrec>
    </carrecs>
</professor>
```

1 Exemple 1: Document XML

```css
professor {
    padding: 30px;
    margin: 30px;
    border: 4px black solid;
    width: 40%;
}

nom,cognom {
    font-size:30px;
}

departament {
    padding-top:20px;
    display:block;
    font-weight:bold;
}

carrec{
    font-style: italic;
    padding-left:10px;
}

carrec:after { content:","; }
```

2 Exemple 1: Document CSS

El resultat és mostrarà formatat:

3 Exemple 1: Resultat de presentació

Malgrat això, CSS té **moltes limitacions** a l'hora de presentar la informació:

- La informació no pot ser reordenada con vulguem. L'única manera de simular el canvi d'ordre és fer servir posicionaments absoluts.
- Els atributs es poden mostrar però hi ha moltes limitacions per fer-ho.
- No es poden afegir estructures noves producte de càlculs o de procés de les dades del document.
- No té maneres senzilles de formatar les dades en pàgines de text per ser impreses.

Si l'objectiu final no és simplement decorar el fitxer sinó transformar-lo en un altre document totalment diferent, CSS no serveix. CSS no transforma el document sinó que simplement canvia la manera com es visualitza.

2. Transformació de documents

Per intentar aconseguir fer tot allò que CSS no podia fer es va crear un nou llenguatge de plantilles**: XSL (extensible stylesheet language).**

Inicialment es van concentrar a poder representar la informació de manera que pogués ser mostrada en documents impresos i en pantalla però després es va acabar definint **un sistema per fer transformacions genèriques de documents XML en altres coses: documents de text, documents XML, pàgines web**, etc.

Actualment XSL és una família de llenguatges que serveixen per definir transformacions i presentacions de documents XML.

La família XSL està formada per tres llenguatges:

- **XSL-FO (XSL formatting objects):** un llenguatge per definir el format que s'ha d'aplicar a un document.
- **XSLT (XSL transformations):** un llenguatge per transformar documents XML.
- **XPath**: un llenguatge per accedir a parts dels documents XML.

2.1. XSL-FO

XSL-FO és un llenguatge basat en XML que està pensat per donar format als documents XML, però a diferència d'altres llenguatges amb objectius similars, com CSS o XHTML, està pensat per generar sortides tant per a formats de pantalla com per a formats paginats.

XSL-FO és un llenguatge que:

- Permet especificar amb molta precisió el contingut d'un document (paginació, organització, estil, etc.).
- Permet crear documents d'alta qualitat.
- És ideal per generar documents amb dades que canvien sovint.

XSL-FO sobretot es fa servir per generar documents en formats pensats per ser impresos com **PDF** o **Postcript**.

Un dels processadors d'XSL-FO més populars és **l'Apache FOP**, que permet crear documents en PDF a partir de documents XSL-FO. Podem obtindré més informació d'aquest processador al següent enllaç: https://xmlgraphics.apache.org/fop/.

És corrent que la generació del document es faça en dues fases:

1. Transformació del document XML en un document XSL-FO amb el llenguatge de transformacions XSLT.
2. Transformació del document XSL-FO en el format que volem amb un processador XSL-FO.

2.2. Processadors XPath o XSLT

En general els processadors XPath o XSLT es proporcionaran per mitjà de biblioteques que podran ser cridades des dels programes.

Això permet simplificar tot el procés de creació de programes que facen servir XPath o XSLT, ja que quan una aplicació es vulga aprofitar de les característiques d'algun d'aquests llenguatges no ho haurà d'implementar tot de nou sinó simplement fer servir les biblioteques.

3. XPath

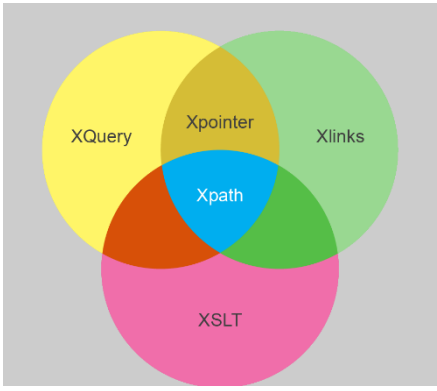
XPath (XML path language) és una manera d'especificar parts d'un document XML que té eines per manipular el contingut de les dades de text, numèriques, etc.

XPath és una recomanació del W3C (www.w3.org/TR/xpath), que tot i que serveix per treballar amb XML no és un llenguatge XML. La idea és que en no estar basat en XML es podrà incloure en altres llenguatges XML sense haver de preocupar-se de si el resultat està ben format o no.

La base del funcionament d'XPath és l'avaluació d'expressions. Una expressió que s'avaluarà contra un document XML i ens donarà un resultat que pot ser de diferents tipus:

- Un booleà: cert o fals
- Un nombre
- Una cadena de caràcters
- Un grup d'elements

XPath està desenvolupat pels comitès de creació d'XSL i XQuery i s'ha convertit en un component essencial per a diferents llenguatges XML com XLinks, XSLT i XQuery, com es pot veure en la següent figura 4:



4 Relació d'XPath amb els llenguatges XML

La versió 2.0 d'XPath està tan integrada dins d'XQuery que qualsevol expressió XPath és també automàticament una expressió XQuery correcta.

3.1. Vista d'arbre

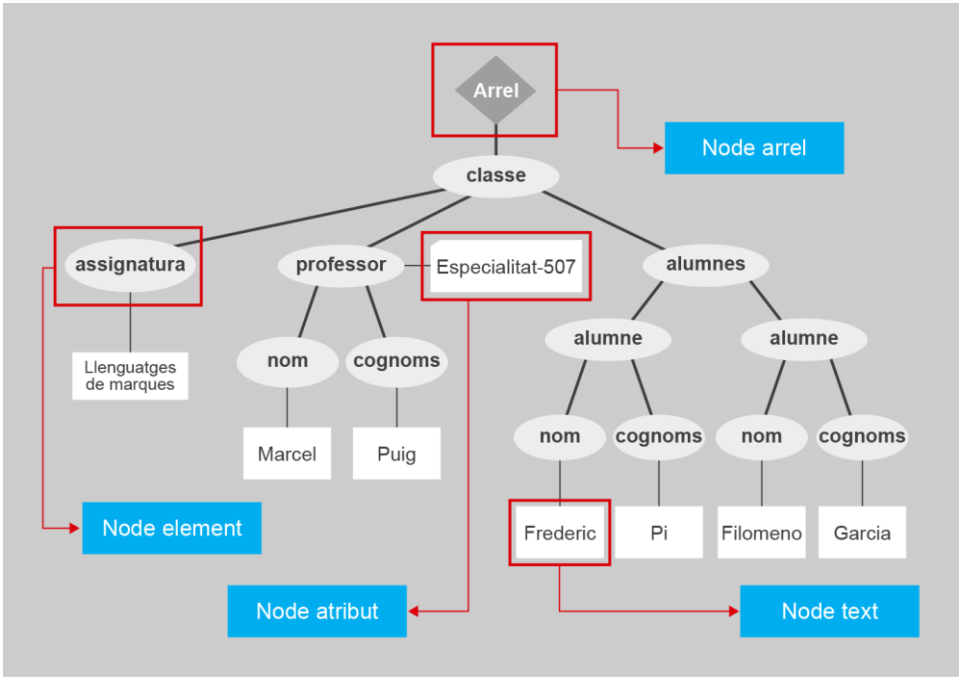
XPath tracta tots els documents XML des del punt de vista **d'un arbre de nodes** en què hi ha una **arrel** que no es correspon amb l'arrel del document, sinó que **és el document** i es representa amb el símbol **"/".**

A part de l'arrel també hi ha nodes per representar els **elements**, els **atributs**, els nodes de **dades**, els **comentaris**, les **instruccions de procés** i els **espais de noms**.

L'exemple següent XML:

```xml
<?xml version="1.0" ?>
<classe>
    <assignatura>Llenguatges de marques</assignatura>
    <professor Especialitat="507">
        <nom>Marcel</nom>
        <cognoms>Puig</cognoms>
    </professor>
    <alumnes>
        <alumne>
            <nom>Frederic</nom>
            <cognoms>Pi</cognoms>
        </alumne>
        <alumne>
            <nom>Filomeno</nom>
            <cognoms>Garcia</cognoms>
        </alumne>
    </alumnes>
</classe>
```

Es representarà en XPath amb un arbre com el següent:



5 Representació en forma d'arbre XPath de l'XML d'exemple

En un arbre XPath els atributs no són considerats nodes fills sinó que són "**propietats**" del node que els conté i els nodes de dades són nodes **sense nom** que només contenen les dades.

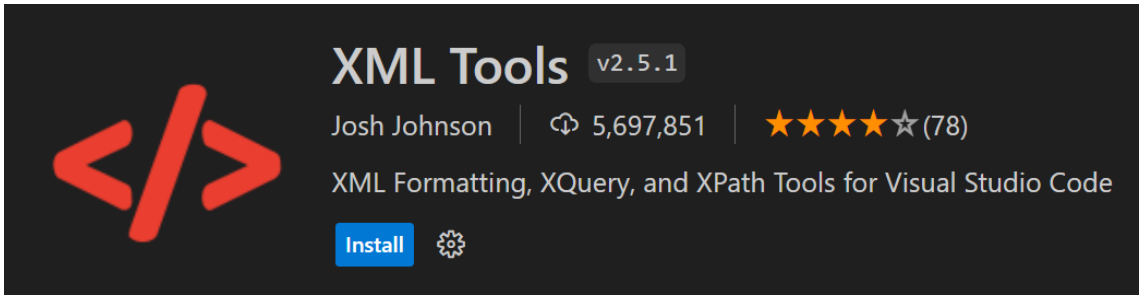
3.2. Programari per avaluar XPath

Molts programes permeten executar una consulta XPath contra un document XML. Hi ha programes específics, editors XML, components dels navegadors web, pàgines web online, etc.

3.2.1. Visual Studio Code

Podem utilitzar l'editor que hem fet servir en altres unitats d'aquest curs, Visual Studio Code, per avaluar expressions XPath fent servir extensions.

Una de les extensions més popular per dur a terme aquesta tasca és XML Tools, de l'autor Josh Johnson



3.2.2. XML Copy Editor

XML Copy Editor és un editor de documents XML lliure (Llicència GPL 2.0) i multiplataforma la pàgina web de la qual és https://xml-copy-editor.sourceforge.io/.

L'última versió disponible actualment (abril de 2024) és la versió XML Copy Editor 1.3.1.0 (del 8 d'octubre de 2022).

Algunes de les seues característiques més destacades són:

- Validació automàtica mentre escrius
- Protecció d'etiquetes
- Suport per a XML Schema/Relax NG/DTD
- Suport per a XSLT i XPath

Pots consultar la secció d'XML Copy Editor a mclibre on s'explica com instal·lar i configurar aquest editor: https://www.mclibre.org/consultar/xml/otros/xmlcopyeditor.html

3.3. Navegació

Com que la representació interna del document XML per XPath serà un arbre, es pot navegar especificant camins d'una manera semblant a com es fa en els directoris dels sistemes operatius.

El més important per tenir en compte a l'hora de crear una expressió XPath és saber el node en el qual està situat el procés (**node de context**), ja que és des d'aquest que s'avaluarà l'expressió. El node de context al principi és en l'arrel però es va movent a mesura que es van avaluant les expressions, i per tant podem expressar els camins XPath de dues maneres:

- Camins absoluts
- Camins relatius

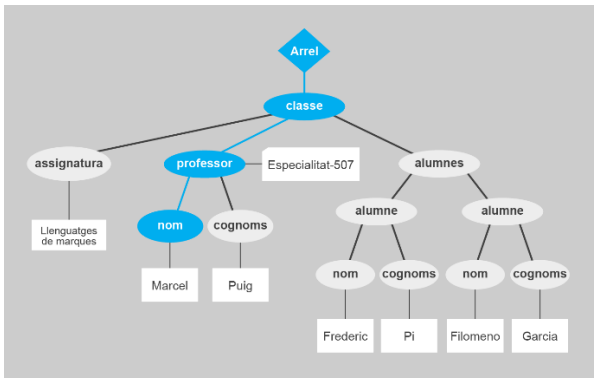
Els **camins absoluts** són camins que sempre comencen en l'arrel de l'arbre. Es poden identificar perquè el primer caràcter de l'expressió sempre serà **l'arrel "/".** No importa quin siga el node de context si es fan servir camins absoluts, perquè el resultat sempre serà el mateix.

En canvi, els **camins relatius** parteixen des del node en el qual estem situats.

Per exemple, es pot obtenir el node <nom> del professor de l'exemple que hem especificat anteriorment fent servir l'expressió XPath següent:

/classe/professor/nom

Podem veure com s'avalua l'expressió en l'arbre XPath en la següent figura:



6 Avaluació de l'expressió /classe/professor/nom

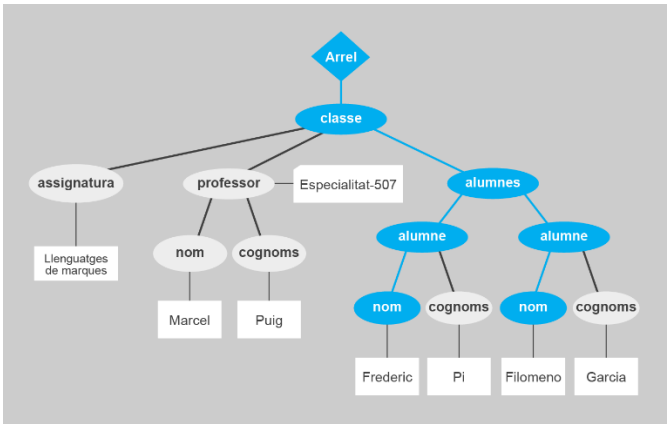
El resultat d'aquesta expressió no és només el contingut de l'element sinó tot l'element <nom>:

<nom>Marcel</nom>

L'expressió sempre intenta aconseguir **el nombre màxim de camins correctes** i, per tant, no necessàriament ha de retornar només un sol valor. Per exemple, si l'expressió per avaluar fos la següent:

/classe/alumnes/alumne/nom

XPath l'avaluaria intentant aconseguir tots els camins que quadren amb l'expressió, tal com podeu veure a la següent figura:



7 Avaluació de l'expressió /classe/alumnes/alumne/nom

El resultat seran els dos resultats possibles:

<nom>Frederic</nom>

<nom>Filomeno</nom>

XPath pot retornar qualsevol tipus d'element com a resultat, no cal que siguen nodes finals. La següent expressió:

/classe/alumnes

retornarà el resultat següent:

```
<alumnes>
    <alumne>
        <nom>Frederic</nom>
        <cognom>Pi</cognom>
    </alumne>
    <alumne>
        <nom>Filomeno</nom>
        <cognom>Garcia</cognom>
    </alumne>
</alumnes>
```

Si se sap que una expressió retornarà diversos resultats però només se'n vol un d'específic es pot fer servir un nombre envoltat per claudàtors quadrats "[]" per indicar quin és el que es vol aconseguir. Per retornar només el primer alumne podeu fer el següent:

/classe/alumnes/alumne[1]

Dels dos nodes disponibles com a fills de <alumnes>, només se seleccionarà el primer:

```
<alumne>
   <nom>Frederic</nom>
   <cognom>Pi</cognom>
</alumne>
```

Obtenir els atributs d'un element

Els valors dels atributs es poden aconseguir especificant el símbol @ davant del nom una volta s'haja arribat a l'element que el conté.

Per exemple, l'expressió XPath:

`/classe/professor/@especialitat`

retornarà:

`507`

S'ha de tenir en compte que a diferència del que passa amb els elements, en obtenir un atribut no tindrem un element sinó **només el seu valor**.

Obtenir el contingut d'un element

Per a aquells casos en què només vulguem el contingut de l'element, s'ha definit la funció text() per obtenir aquest contingut. Això s'ha fet així perquè d'altra manera, com que els nodes de text no tenen nom, no es podria accedir al seu contingut.

De manera que si a un element que tinga contingut de dades se li afegeix text:

```
/classe/professor/nom/text()
```

retornarà el contingut del node sense les etiquetes:

```
Marcel
```

Comodins

De la mateixa manera que en els sistemes operatius, es poden fer servir comodins diversos en les expressions XPath.

Comodí	**Significat**
*	L'asterisc es fa servir per indicar tots els elements d'un determinat nivell.
.	Com en els directoris dels sistemes operatius el punt serveix per indicar el node actual.
..	Es fa servir per indicar el pare del node en el qual estem.
/ /	Les dobles barres indiquen que quadrarà amb qualsevol cosa des del node en el qual estem. Pot ser un sol element o un arbre de nodes.

Eixos XPath

Per avaluar les expressions XPath s'explora un arbre, de manera que també es proporcionen una sèrie d'elements per fer referència a parts de l'arbre. Aquests elements s'anomenen eixos XPath.

Eix	**Significat**
self::	El node en el qual està el procés (fa el mateix que el punt)
child::	Fill de l'element actual
parent::	El pare de l'element actual (idèntic a fer servir ..)
attribute::	Es fa servir per obtenir un atribut de l'element actual (@)
descendant::	Tots els descendents del node actual
desdendant-or-self::	El node actual i els seus descendents
ancestor::	Els ascendents del node
ancestor-or-self::	El node actual i els seus ascendents
prededint::	Tots els elements precedents al node actual
preceding-sibling::	Tots els germans precedents
following::	Elements que segueixen el node actual
following-sibling::	Germans posteriors al node actual
namespace::	Conté l'espai de noms del node actual

Condicions

Un apartat interessant de les expressions XPath és poder afegir condicions per a la selecció de nodes. A qualsevol expressió XPath se li poden afegir condicions per obtenir només els nodes que compleixin la condició especificada.

La selecció de nodes es fa especificant un predicat XPath dins de claudàtors.

Per exemple, aquesta expressió selecciona només els professors que tinguen un element <nom> com a fill de <professor>:

/classe/professor[nom]

Si s'aplica l'expressió a l'exemple que hem fet servir per fer la vista d'arbre, el resultat serà el node <professor> que té dins seu <nom>.

```
<professor especialitat="507">
  <nom>Marcel</nom>
  <cognoms>Puig</cognoms>
</professor>
```

En el valor de l'expressió s'especifiquen camins relatius des del node que tinga la condició. Fent servir condicions es pot fer una expressió que només retorne el professor si té alumnes.

/classe/professor[../alumnes/alumne]

Normalment la complexitat de les condicions va més enllà de comprovar si el node existeix, i es fan servir per comprovar si un node té un valor determinat. Per exemple, per obtenir els professors que es diguen "Marcel":

/classe/professor[nom="Marcel"]

Les condicions es poden posar en qualsevol lloc del camí i pot haver tantes com calga. Per obtenir el cognom del professor que es diu "Marcel" es pot fer servir una expressió com aquesta.

/classe/professor[nom="Marcel"]/cognoms

Que donarà de resultat:

<cognoms>Puig</cognoms>

3.4. Seqüències

Una seqüència és una expressió XPath que retorna més d'un element. S'assemblen bastant a les llistes d'altres llenguatges:

- Tenen ordre
- Permeten duplicats
- Poden contenir valors de tipus diferent en cada terme

És fàcil crear seqüències, ja que només cal tancar-les entre parèntesi i separar cada un dels termes amb comes. L'expressió següent aplicada a qualsevol document:

(1,2,3,4)

Retorna la seqüència de nombres d'un en un:

```
1
2
3
4
```

També es poden crear seqüències a partir d'expressions XPath. En aquest cas s'avaluarà primer la primera expressió, després la segona, etc.

```
(//nom/text(), //cognoms/text())
```

Aplicat al nostre exemple retornarà primer tots els noms i després tots els cognoms:

```
Marcel
Frederic
Filomeno
Puig
Pi
Garcia
```

Unió, intersecció i disjunció

També es pot operar amb les seqüències d'elements. Una manera seria fer servir els operadors d'unió (union), intersecció (intersec) o disjunció (except).

Per exemple, l'expressió següent ens retornaria els cognoms dels alumnes que coincideixin amb els d'un professor:

```
(//alumne/nom) intersect (//professor/nom)
```

Amb la unió es poden unir les llistes de manera que en quedi una de sola sense duplicats:

```
(//alumne/nom) union (//professor/nom)
```

I amb la disjunció obtenim els noms de la primera seqüència que no surten en la segona:

```
(//alumne/nom) except (//professor/nom)
```

3.5. Funcions

XPath ofereix una gran quantitat de funcions destinades a manipular els resultats obtinguts.

Les funcions que ofereix XPath es classifiquen en diferents grups:

- Per manipular conjunts de nodes: es pot obtenir el nom dels nodes, treballar amb les posicions, comptar-los, etc.
- Per treballar amb cadenes de caràcters: permeten extreure caràcters, concatenar, comparar… les cadenes de caràcters.
- Per fer operacions numèriques: es poden convertir els resultats a valors numèrics, comptar els nodes, fer-hi operacions, etc.
- Funcions boleanes: permeten fer operacions boleanes amb els nodes.

- Funcions per dates: permeten fer operacions diverses amb dates i hores.

És impossible especificar-les totes aquí, de manera que el millor és consultar l'especificació XPath (http://www.w3.org/TR/xpath-functions) per veure quines funcions es poden fer servir.

Amb les funcions es podran fer peticions que retornen valors numèrics. Per exemple, "quants alumnes tenim?":

```
count(/classe/alumnes/alumne)
```

Que tornarà el nombre d'alumnes del fitxer:

2

O bé retornar cadenes de caràcters. Per exemple, unir tots els noms separant-los per una coma:

```
string-join(//nom,",")
```

Que retornarà en un únic resultat els noms separats per una coma:

```
Marcel,Frederic,Filomeno
```

També es poden posar les funcions en els predicats. Per exemple, aquesta expressió ens retornarà els alumnes amb cognom que comenci per p.

```
/classe/alumnes/alumne[starts-with(cognoms,"P")]
```

En aquesta expressió volem obtenir el segon alumne de la llista:

```
/classe/alumnes/alumne[position()=2]
```

4. XSLT

XSLT (extensible stylesheet language for transformations) és un **llenguatge de plantilles basat en XML** que permet convertir l'estructura dels elements XML en altres documents.

Amb XSLT es poden fer transformacions que canvien totalment l'estructura del document, pot reordenar la informació del document XML, afegir informació nova on siga, prendre decisions en funció de la informació que es trobe, fer càlculs, etc.

XSLT es recolza en altres tecnologies XML per funcionar:

- Fa servir XPath per determinar les plantilles per aplicar en cada moment (i per tant s'integra en XQuery).
- Suporta XML Schemas per definir els tipus de dades.

La versió 2.0 de XSLT afegeix tota una sèrie de característiques que encara fan més potent XSLT:

- Suporta els tipus de dades d'XML Schemas
- Inclou elements nous que permeten agrupar resultats, etc.
- Pot generar múltiples eixides en una sola transformació
- Afegeix un grup de funcions noves a més de les d'XPath 2.0
- Pot processar fitxers que no siguen XML

Una de les **limitacions** d'XSLT és que el document d'entrada ha de ser sempre XML, a pesar que la sortida no té aquesta limitació i pot acabar essent en qualsevol format: text, HTML, XML, etc.

4.1. Programari XSLT

Uns dels programes d'ús corrent que tenen incorporades biblioteques XSLT són els navegadors web (Firefox, Internet Explorer, Google Chrome, i d'altres). Si es carrega un document XML que té associada una transformació XSLT en un navegador web es transformarà automàticament i es mostrarà el resultat transformat.

Les plantilles XSLT són documents XML que es poden crear fent servir editors de text o bé editors especialitzats en XML. En la creació de plantilles XSLT, els avantatges que ofereixen els editors XML són tan importants (depuradors d'expressions, ajudes…) que els fan una eina molt valuosa per treballar professionalment.

Tal i com ja s'ha vist a la secció de programari d'Xpath, podem utilitzar tant Visual Studio Code (extensió XML Tools) com XML Copy Editor per definir i executar transformacions XSLT.

4.2. El procés de transformació

Una transformació XSLT consisteix a passar un document XML i una plantilla per un processador XSLT, el qual aplicarà les regles que trobe en la plantilla al document per generar un document nou. El fitxer de resultat de la transformació pot ser tant un document XML com en qualsevol altre format.

Per associar un document XML a una plantilla XSLT específica es fa servir l'etiqueta <?xml-stylesheet ?>.

Per exemple si volem associar la plantilla XSLT alumnes.xsl a un document cal editar-lo i afegir-hi l'etiqueta <?xml-stylesheet ?> rere la declaració xml:

```
<?xml-stylesheet href="alumnes.xsl" type="text/xsl" ?>
```

L'arrel del document XSLT

Les plantilles XSLT són documents XML, i per tant, han de complir les regles dels documents XML. Això vol dir que han de tenir un element arrel.

L'arrel de les plantilles XSLT és l'element <stylesheet>.

L'element arrel <stylesheet> ha de tenir obligatòriament dos atributs:

- L'atribut version, que especificarà quina és la versió d'XSLT que s'ha fet servir per crear la plantilla i que es farà servir per fer la transformació.
- L'espai de noms de XSLT, que normalment s'associa amb l'àlies 'xsl'.

Per exemple, per a la versió 2.0 d'XSLT la definició de l'arrel seria una cosa com aquesta:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  version="2.0">
</xsl:stylesheet>
```

Dins de l'arrel es definiran les plantilles que determinaran quina és la transformació que es vol fer en el document original.

Crear una plantilla

El procés de transformació es farà intentant aplicar alguna plantilla als nodes del document original.

Qualsevol element que no es puga processar amb una plantilla es transformarà amb el funcionament per defecte:

- Si el node té contingut es retornarà el contingut del node
- Si el node no té contingut es retornarà sense escriure res

Per tant l'element bàsic per fer les transformacions són les plantilles. Les plantilles es defineixen amb l'element **<xsl:template>.**

L'element de plantilla pot tenir diversos atributs però el més corrent és match. L'atribut match serveix per determinar a quins nodes s'ha d'aplicar aquesta plantilla per mitjà d'una expressió XPath.

En el contingut de l'element de plantilla s'especificarà quina és la transformació que cal aplicar als elements. La transformació més senzilla consisteix a escriure un text literal. Per exemple, en la plantilla següent s'està definint que en arribar algun element <nom> es desi en el fitxer de destinació la lletra A:

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

Per tant si s'aplica la plantilla anterior a aquest document:

```
<persona>
    <nom>Pere Garcia</nom>
</persona>
```

El processador anirà analitzant els nodes del document original i primer intentarà buscar una plantilla per a l'element <persona>, i com que no en trobarà cap recorrerà al comportament per defecte i escriurà una línia en blanc.

Després intentarà trobar una plantilla per a l'element <nom> que, segons la plantilla, s'ha de transformar en una lletra A.

Com que no hi ha més nodes el resultat serà aquest:

```
_
    A
```

Cal tenir en compte que un cop un element ha estat processat per una plantilla se'n processa tot el seu contingut amb ell (i, per tant, també els elements que puga contenir). Si canviem la plantilla anterior per la següent:

```
<xsl:template match="persona">
    Persona
</xsl:template>
<xsl:template match="nom">
    A
</xsl:template>
```

En aplicar-la al mateix XML primer processarà l'element <persona> i el transformarà en "Persona" en el document de sortida, però com que <persona> conté tots els altres elements, la transformació s'acabarà.

```
Persona
```

Afegir elements

El fet de poder fer transformacions fent servir text literal fa que també es puga fer servir el mateix sistema per crear etiquetes noves. Si es modifica la plantilla anterior per una com la següent:

```
<xsl:template match="nom">
    <senyor>Manel</senyor>
</xsl:template>
```

El resultat serà que el transformarà en aquest:

```
<senyor>Manel</senyor>
```

S'ha d'anar amb compte quan s'especifiquen etiquetes literals, ja que no es poden crear etiquetes que deixin la plantilla mal formada. La plantilla següent que només obre l'etiqueta <senyor> no es pot fer servir, perquè deixa la plantilla mal formada:

```
<xsl:template match="nom">
    <senyor>Manel
</xsl:template>
```

Una manera alternativa de crear elements en una transformació és fer servir <xs:element>. Entre els atributs que pot tenir, l'únic obligatori és name, que defineix el nom que tindrà l'etiqueta.

Per exemple, amb la plantilla següent creem un element <senyor> que sempre tindrà de contingut "Pere" per a cada element <nom> que es trobi en el document.

```
<xsl:template match="nom">
    <xs:element name="senyor">Pere</xs:element>
</xsl:template>
```

O sigui, que la sortida de cada element <nom> serà:

```
<senyor>Pere</senyor>
```

Afegir atributs

Els atributs es poden definir dins d'un nou element amb <xsl:attribute> i de la mateixa manera que amb els nous elements, l'únic atribut obligatori és name.

Una transformació que tingués això:

```
<xs:element name="persona">
   <xsl:attribute name="home">Si</xsl:attribute>
   Marcel
</xsl:element>
```

Generaria el següent:

```
<persona home="Si">Marcel</persona>
```

Control de les eixides de text

En comptes de fer servir literals també es poden fer transformacions en text fent servir l'element **<xsl:text>.** Aquest element és important quan es volen controlar millor els espais i els salts de línia que hi haurà en les sortides.

Podem fer que isquen 5 espais darrere de la lletra A definint la plantilla de la manera següent:

```
<xsl:template match="nom">
    <xsl:text>A     </xsl:text>
</xsl:template>
```

Si no ho ferem amb <xsl:text> els espais de darrere la A es perdrien.

Obtenir valors

Una tasca habitual a l'hora de fer una transformació sol ser obtenir els valors dels elements del document d'origen per posar-los en l'element de destinació. Es poden obtenir el valor dels elements d'origen fent servir **<xsl:value-of>.**

Aquest element avalua l'expressió XPath de l'atribut select i genera una sortida amb el seu contingut.

Per exemple, a partir de l'XML següent:

```
<xsl:template match="persona">
    <xsl:value-of select="nom">
</xsl:template>
```

Afegirà el valor de l'element <nom> en el fitxer d'eixida:

```
Pere Garcia
```

Però s'ha de tenir en compte que només avalua el primer element que compleixi la condició, o siga, que si el fitxer de mostra fora el següent:

```
<?xml version="1.0" ?>
<persona>
    <nom tipus="Professor">Pere Garcia</nom>
    <nom>Frederic Pi</nom>
    <nom>Manel Puig</nom>
</persona>
```

…l'eixida d'aplicar la mateixa plantilla només seria el primer dels noms i no tots dos, perquè agafaria només el primer <nom> de <persona>:

```
Pere Garcia
```

Per tant és molt important triar bé quines són les expressions XPath de les plantilles per evitar aquest comportament. En aquest cas la solució podria ser escollir <nom> en comptes de <persona>:

```
<xsl:template match="nom">
    <xsl:value-of select="."/>
</xsl:template>
```

Com a resultat de l'element <xsl:value-of> s'hi pot posar qualsevol expressió XPath. Per tant, també es poden obtenir els valors dels atributs afegint "@" davant del nom.

```
<xsl:template match="nom">
    <xsl:value-of select="."/> ( <xsl:value-of select="@tipus" /> )
</xsl:template>
```

Que generarà:

```
Pere Garcia ( Professor )
Frederic Pi (  )
Manel Puig (  )
```

I també es poden fer servir les funcions XPath. Per tant, es pot generar una sortida amb el total de les persones del fitxer:

```
<xsl:template match="/">
    Total: <xsl:value-of select="count(//nom)"/>
</xsl:template>
```

Que donarà de resultat:

```
Total: 3
```

Reenviar nodes a una altra plantilla

Un altre ús habitual de les plantilles és el de fer-les servir per cridar-ne d'altres quan es volen processar alguns dels nodes obtinguts per la plantilla quan aquesta n'obté diversos.

Les crides a plantilles es poden fer amb l'element <xsl:apply-templates>. Amb aquest element es pot fer que el diferents resultats d'una expressió XPath es vagin processant un a un, buscant una plantilla on aplicar-se.

Si partim d'aquest codi XML:

```
<classe>
    <nom>Frederic Pi</nom>
    <nom>Filomeno Garcia</nom>
    <nom>Manel Puigdevall</nom>
</classe>
```

I el full de plantilles següent:

```
<xsl:template match="/">
    <xsl:apply-templates select="classe/nom"/>
</xsl:template>
<xsl:template match="nom">
    <xsl:value-of select="."/>
</xsl:template>
```

Al executar-se l'apply-templates s'avalua l'expressió XPath que conté, classe/nom, que retorna tres resultats:

```
<nom>Frederic Pi</nom>
<nom>Filomeno Garcia</nom>
<nom>Manel Puigdevall</nom>
```

Cada un d'aquests resultats individualment intentarà trobar una plantilla on aplicar-se. De manera que es processarà tres vegades el segon <xsl:template>, ja que és qui captura nom, i donarà com a resultat:

```
Frederic Pi
Filomeno Garcia
Manel Puigdevall
```

Una de les característiques interessants és que <apply-templates> també es pot fer servir per reordenar el contingut. Per exemple, amb l'XML següent:

```
<classe>
   <professors>
       <nom>Marcel Puig</nom>
       <nom>Maria Sabartés</nom>
```

```
        </professors>
        <alumnes>
            <nom>Frederic Pi</nom>
            <nom>Filomeno Garcia</nom>
            <nom>Manel Puigdevall</nom>
        </alumnes>
    </classe>
```

Es poden obtenir tots els nodes amb una plantilla que capturi l'arrel i que primer mostri els noms dels alumnes i després els dels professors, de la següent manera:

```
    <xsl:template match="classe">
        Alumnes
        ----------
        <xsl:apply-templates select="alumnes/nom"/>
        Professors
        ----------
        <xsl:apply-templates select="professors/nom"/>
    </xsl:template>
    <xsl:template match="nom">
        <xsl:value-of select="."/>
    </xsl:template>
```

Que generarà:

```
Alumnes
----------
Frederic Pi
Filomeno Garcia
Manel Puigdevall
Professors
----------
Marcel Puig
Maria Sabartés
```

Es pot veure que un dels usos d'aquest element pot ser reordenar els resultats. En l'exemple, els alumnes en el document original estaven darrere dels professors, i en canvi en el que hem obtingut estan davant.

Processar els nodes per fer tasques diferents

A vegades pot interessar processar els mateixos nodes diverses vegades per obtenir diferents resultats. Per poder fer això es poden definir les plantilles que tinguin l'atribut mode.

Si es defineix una plantilla amb l'atribut mode per activar la plantilla no n'hi haurà prou que quadri amb l'atribut match, sinó que també caldrà que s'hi passi l'atribut mode.

```
<template match="nom" mode="resultat">
    ...
</template>
```

Les plantilles amb l'atribut mode han de ser cridades específicament. Per exemple, amb <apply-templates>:

```
<xsl:apply-templates select="nom" mode="resultat"/>
```

Això permet fer dues coses diferents amb l'element <alumnes> de l'exercici anterior: comptar els alumnes i fer-ne una llista.

```
<xsl:template match="classe">
    <xsl:apply-templates select="//alumnes"
mode="resultat"/>
    <xsl:apply-templates select="//alumnes"/>
</xsl:template>
<xsl:template match="alumnes">
    <xsl:apply-templates select="nom" />
</xsl:template>
<xsl:template match="alumnes" mode="resultat">
    Total: <xsl:value-of select="count(nom)"/>
</xsl:template>
<xsl:template match="nom">
    <xsl:value-of select="."/>
</xsl:template>
```

El resultat serà:

```
Total: 3
Frederic Pi
Filomeno Garcia
Manel Puigdevall
```

Forçar el tipus d'eixida

XSLT està pensat per generar eixides en text, XML, HTML i XHTML. Per defecte considera que l'eixida serà un altre document XML i, per tant, si es vol generar un document XML no cal especificar res.

Podem forçar que l'eixida siga una altra amb <xsl:output>. Per exemple podem fer que la sortida sigui interpretada com a text pla amb:

```
<xsl:output type="text">
```

Si l'atribut type no és "xml" no sortirà la capçalera XML en el resultat final.

Però a part de type hi ha altres atributs que permeten controlar de quina manera es generaran els resultats. Per exemple, **indent** serveix per formatar les eixides i **encoding** per definir la codificació.

Instruccions de control

XSLT proporciona una manera de processar els resultats molt semblant a com ho fan els llenguatges de programació. Com molts llenguatges de programació, incorpora instruccions per processar els resultats.

Expressions condicionals

<xsl:if> permet afegir els resultats al fitxer només si es compleix una determinada condició.

Per exemple, davant d'un document com el següent:

```
<alumnes>
    <alumne nota="5">Pere Garcia</nota>
    <alumne nota="3">Manel Puigdevall</nota>
</alumnes>
```

Amb <xsl:if> es pot aconseguir posar algun text només en els alumnes que tinguin una nota superior a 5.

```
<xsl:if test="@nota>=5">
   (aprovat)
</xsl>
```

Hi ha altres funcions per expressar alternatives, com **<xsl:choose>,** que permet explicitar múltiples alternatives.

```
<xsl:choose>
    <xsl:when select="@nota=10"> (excel·lent) </xsl:when>
    <xsl:when select="@nota>=5"> (aprovat)    </xsl:when>
    <xsl:otherwise> (suspès) </xsl:otherwise>
</xsl:choose>
```

Iteracions

<xsl:for-each> serveix per processar una seqüència de nodes un per un per fer alguna acció en cada un:

```
<xsl:for-each select="/classe/alumnes/nom">
    <xsl:value-of select="."/>
</xsl:for-each>
```

5. Enllaços d'interés

XSLT a **w3schools**: https://www.w3schools.com/xml/xsl_intro.asp

Especificació XSLT 1.0 a w3: www.w3.org/TR/xslt

Especificació XSLT 2.0 a w3: www.w3.org/TR/xslt20/

Xpath a **w3schools**: https://www.w3schools.com/xml/xpath_intro.asp

Hojas de estilo CSS en XML, **mclibre**: https://www.mclibre.org/consultar/xml/lecciones/xml-css.html

Apache FOP: https://xmlgraphics.apache.org/fop/

6. Bibliografia

Sala, Xavier. (2023) Llenguatges de marques i sistemes de gestió d'informació.

Institut Obert de Catalunya