Llenguatges de Marques i Sistemes de Gestió d'Informació

UD 3.2 Transformacions XSLT

IES Jaume II El Just
Tavernes de la Valldigna
Departament d'Informàtica
Curs 2024-25

Índex

1 3.2. XSL

XSL (Extensible Stylesheet Language "llenguatge extensible de fulles d'estil") és 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 específic.

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 formato 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**, o XML Path Language, és una sintaxi per a accedir o referir-se a porcions d'un document XML.
- **XQuery**,un llenguatge de consulta de documents XML.

Aquestes tres especificacions són recomanacions oficials del W3C.

1.1 XSLT

XSLT s'utilitza per transformar un document XML en un altre document XML o un altre tipus de document reconegut per un navegador, com HTML.

Amb XSLT podem afegir/eliminar elements i atributs al fitxer d'eixida. També podem reordenar i ordenar els elements.

Una manera habitual de descriure el procés de transformació és dir que XSLT transforma un arbre d'origen XML en un arbre de resultats XML.

XSLT utilitza **XPath** per trobar informació en un document XML. XPath s'utilitza per navegar per elements i atributs en documents XML.

En el procés de transformació, XSLT utilitza XPath per definir parts del document font que han de coincidir amb una o més plantilles predefinides. Quan es troba una coincidència, XSLT transformarà la part coincident del document d'origen en el document de resultats.

1.2 Declaració del document XSLT

L'element arrel que declara que el document és un full d'estil XSL és `<xsl:stylesheet>` o `<xsl:transform>`.

> Nota: `<xsl:stylesheet>` i `<xsl:transform>` són completament sinònims i es poden utilitzar indiferentment.

La manera correcta de declarar un full d'estil XSL segons la recomanació XSLT del W3C és:

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

o:

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

En el document XML que volem transformar, hem d'afegir la declaració de quin document xslt volem utilitzar per a fer la transformació:

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

> Els documents xslt han de tindre l'extensió `.xsl`.

1.3 XSLT Template

Un full d'estil XSL consta d'un o més conjunt de regles que s'anomenen *templates* (plantilles).

Una plantilla conté regles per aplicar quan coincideix un node especificat.

L'element `<xsl:template>` s'utilitza per crear plantilles.

Aquest element té un atribut, `match`, que s'utilitza per associar una plantilla amb un element XML. L'atribut `match` també es pot utilitzar per definir una plantilla per a tot el document XML. El valor de l'atribut match és una expressió XPath (és a dir, `match="/"` defineix tot el document). Aquesta és el tipus de plantilla que utilitzarem en aquests apunts, encara que també es poden associar templates a elements concrets.

> Els elements xsl:stylesheet, xsl:transform i xsl:template s'han de tancar amb la corresponent etiqueta de tancament.

Exemple de plantilla genèrica:

```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="/">
  <!-- Ací posem els elements de la plantilla -->
</xsl:template>

</xsl:stylesheet>
```

1.4 Recuperar valors

L'element `<xsl:value-of>` es pot utilitzar per extraure el valor d'un element XML i afegir-lo al flux d'eixida de la transformació.

Per exemple, per al següent xml:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persona>
    <nombre>Pepe</nombre>
    <apellido1>Lopez</apellido1>
    <apellido2>Faus</apellido2>
</persona>
```

Podem donar-li un format en HTML amb la següent plantilla:

```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="/">

        <h1>Persona</h1>
        <p><strong>Nombre: </strong>
          <xsl:value-of select="persona/nombre"/>
        </p>
        <p><strong>Primer apellido: </strong>
          <xsl:value-of select="persona/apellido1"/>
        </p>
        <p><strong>Segundo apellido: </strong>
          <xsl:value-of select="persona/apellido2"/>
        </p>
```

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

obtenim la següent eixida:

```
<h1>Persona</h1>
<p>
   <strong>Nombre: </strong>Pepe
</p>
<p>
   <strong>Primer apellido: </strong>Lopez
</p>
<p>
   <strong>Segundo apellido: </strong>Faus
</p>
```

Si guardem el resultat anterior en un fitxer html i ho visualitzem amb un navegador web, podem veure el resultat com a pàgina web:

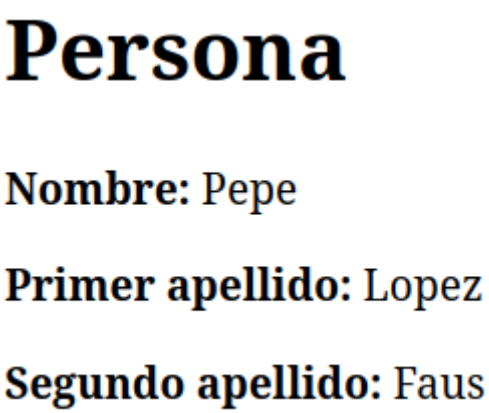

Figura 1: Persona com a HTML

1.5 Utilitzar un parser xsl

Per a fer la transformació xsl necessitem un programa o complement que la puga fer.

Per exemple, en BaseX podem fer les transformacions amb la funció `xslt:transform-text`, en l'apartat file:

```
xslt:transform-text(
  doc("/ruta/fitxer.xml"),
  doc("/ruta/fitxer.xsd")
)
```

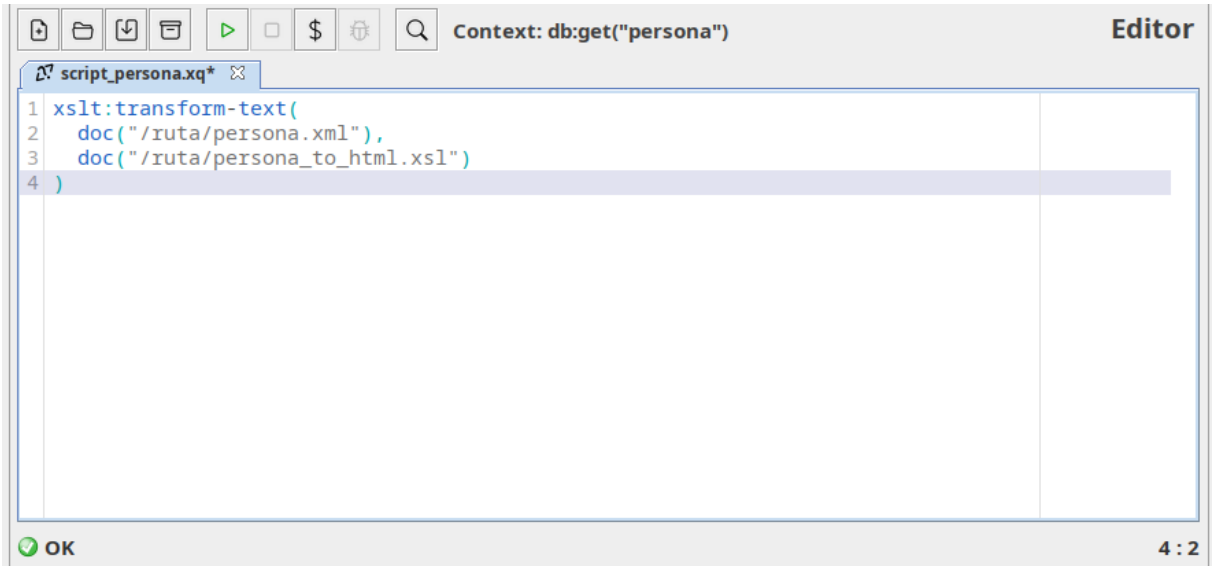

Figura 2: Transformacions

> Nota: reemplaçar `ruta` per la teua ruta local.

L'script generat el podem guardar per a futurs usos.

També podem utilitzar una aplicació web, per exemple:

FreeFormatter.com

1.6 Més exemples amb xsl:value-of

Una altra operació molt comú és transformar un xml en un altre xml.

Per a l'exemple anterior, podem unir els elements `apellido1` i `apellido2` en un únic element anomenat `apellidos`:

```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="/">

        <persona>
            <nombre> <xsl:value-of select="persona/nombre"/> </nombre>
            <apellidos>
                <xsl:value-of select="persona/apellido1"/>
                <xsl:text> </xsl:text>
                <xsl:value-of select="persona/apellido2"/>
            </apellidos>
        </persona>

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

Resultat:

```xml
<persona>
   <nombre>Pepe</nombre>
   <apellidos>Lopez Faus</apellidos>
</persona>
```

> Nota: Hem usat l'element `<xsl:text> </xsl:text>` per a inserir un espai en blanc. Aquest element es pot utilitzar per a escriure qualsevol text, incloent salts de línia.

Un altre exemple seria transformar el fitxer xml en fitxer json:

```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="/">
      {
        "nombre": "<xsl:value-of select="persona/nombre"/>",
        "apellido1": "<xsl:value-of select="persona/apellido1"/>",
        "apellido2": "<xsl:value-of select="persona/apellido2"/>"
      }
    </xsl:template>
</xsl:stylesheet>
```

Resultat:

```
{
  "nombre": "Pepe",
  "apellido1": "Lopez",
  "apellido2": "Faus"
}
```

1.7 Iterar sobre llistes

L'element XSL `<xsl:for-each>` es pot utilitzar per seleccionar tots els elements XML d'un conjunt de nodes especificat:

```
<xsl:for-each select="estudiants/estudiant">
```

El valor de l'atribut select és una expressió XPath que completa la ruta especificada en `xsl:template`. És a dir, si tenim `<xsl:template match="/">`, `<xsl:for-each select="estudiants/estudiant">` equivaldria a l'expressió XPath `/estudiants/estudiant`.

Per exemple, amb l'xml següent:

```
<?xml version="1.0" encoding="UTF-8"?>
<estudiants>
    <estudiant>
        <nom>Josep</nom>
        <cognoms>Faus Gomis</cognoms>
        <edad>18</edad>
    </estudiant>
    <estudiant>
        <nom>Maria</nom>
        <cognoms>Vidal Pérez</cognoms>
        <edad>17</edad>
    </estudiant>
    <estudiant>
        <nom>Anna</nom>
        <cognoms>García López</cognoms>
        <edad>28</edad>
    </estudiant>
    <estudiant>
        <nom>Pau</nom>
        <cognoms>Martínez Sánchez</cognoms>
        <edad>16</edad>
    </estudiant>
</estudiants>
```

I l'arxiu xsl:

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

    <xsl:template match="/" >
        <h1>Estudiants</h1>
        <xsl:for-each select="estudiants/estudiant">
            <p> <strong>Estudiant: </strong>
                <xsl:value-of select="nom"/>
                <xsl:text> </xsl:text>
                <xsl:value-of select="cognoms"/>
            </p>
            <p> <strong>Edat: </strong>
                <xsl:value-of select="edat"/>
            </p>
            <hr/>
        </xsl:for-each>
    </xsl:template>

</xsl:stylesheet>
```

Obtenim:

```
<h1>Estudiants</h1>
<p>
   <strong>Estudiant: </strong>Josep Faus Gomis</p>
<p>
   <strong>Edat: </strong>18</p>
<hr/>
<p>
   <strong>Estudiant: </strong>Maria Vidal Pérez</p>
<p>
   <strong>Edat: </strong>17</p>
<hr/>
<p>
   <strong>Estudiant: </strong>Anna García López</p>
<p>
   <strong>Edat: </strong>28</p>
<hr/>
<p>
   <strong>Estudiant: </strong>Pau Martínez Sánchez</p>
<p>
   <strong>Edat: </strong>16</p>
<hr/>
```

Figura 3: Estudiant en html

Utilitzant **predicats** podem filtrar els resultats de `xsl:for-each`. Per exemple, amb:

```
<xsl:for-each select="estudiants/estudiant[edat>=18]">
```

obtenim només els estudiants Josep Faus Gomis i Anna García López.

També podem ordenar l'eixida amb `<xsl:sort>`:

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

    <xsl:template match="/" >
        <h1>Estudiants</h1>
        <xsl:for-each select="estudiants/estudiant">
        <xsl:sort select="cognoms" />
            <p> <strong>Estudiant: </strong>
                <xsl:value-of select="nom"/>
                <xsl:text> </xsl:text>
                <xsl:value-of select="cognoms"/>
            </p>
            <p> <strong>Edat: </strong>
                <xsl:value-of select="edat"/>
            </p>
            <hr/>
        </xsl:for-each>
    </xsl:template>

</xsl:stylesheet>
```

En aquest exemple hem ordenat per `cognoms`.

També podem ordenar per ordre descendent:

```
<xsl:sort select="cognoms" order="descending" />
```

1.8 Controlant l'eixida

Podem controlar el format d'eixida amb l'element `<xsl:output>`. Aquest és un element d'etiqueta única i que s'ha de posar preferentment a continuació de `<xsl:template>`. Alguns exemples:

```
<xsl:output method="xml" indent="yes"/> exporta a xml amb indentació
<xsl:output method="html" indent="yes"/> exporta a html amb indentació
<xsl:output method="text" indent="no"/> exporta a format text sense
↪  indentació
```

Més sobre xsl:output.

1.9 Resum

- `<xsl:template match="/"> </xsl:template>`: Especifica la plantilla.
- `<xsl:value-of select="expressio_xpath"/>`: Selecciona el valor de l'etiqueta que es correspon amb `expressio_xpath`.
- `<xsl:text> </xsl:text>`: Escriu el contingut entre aquestes etiquetes.
- `<xsl:for-each select="expressio_xpath"> </xsl:for-each>`: Itera sobre tots els elements es corresponen amb `expressio_xpath`.
- `<xsl:sort select="element" order="ascending|descending"/>`: Ordena segons els valors de `element` en ordre ascendent o descendent (per defecte ascendent).
- `<xsl:output method="xml|html|text" indent="yes|no"/>`: Especifica el format d'eixida.

1.10 Referències

- Recomanació XSLT.
- W3Schools XSLT tutorial