XSL

XSLT - XPATH

1.- Introducció

XSL (eXtensible Stylesheet Language):

XSL és un **llenguatge de disseny** específicament creat per a treballar amb **documents XML**.

El seu objectiu principal **és descriure la presentació del contingut XML i la seva estructura**. Amb XSL, es poden definir estils i formats per a les dades contingudes en un document XML, permetent **una separació clara entre el contingut i la seva representació visual**.

Dins del context **d'XSL**, hi ha dues parts essencials:

1. **XSLT (Transformacions XSL):**
    - **XSLT** és una part crucial d'**XSL** i ofereix la capacitat de transformar documents XML en altres formats, com ara HTML o XML modificat.
    - Amb **XSLT**, es poden realitzar diverses manipulacions als continguts **XML**, com ara **ordenar-los**, **filtrar-los** o **afegir estils** per a millorar la presentació.

2. **XPATH (Navegació i Obtenció de Dades):**
    - **XPATH** és un idioma utilitzat per a la **navegació i obtenció de dades en documents XML**.
    - S'utilitza per especificar la ubicació exacta d'elements o atributs dins d'un document XML mitjançant una sintaxi similar a les rutes d'accés als sistemes de fitxers.

Amb **XSLT** i **XPATH** podem generar un **document HTML** a partir de les dades contingudes en un **fitxer XML**. Aquest procés de transformació permet adaptar les dades a un format visual adequat per a la presentació, separant la lògica de presentació dels continguts del document XML.

Usant estes eines, és possible realitzar operacions avançades, com seleccionar nodes específics i filtrar dades del document l'XML, i sobre tot, crear documents HTML estructurats bastes el la informació continguda en eixos XML.

Este procés facilitarà la creació de pàgines WEB dinàmiques i altament personalitzades a partir de dades en format XML

XPATH – Sintaxi

```xml
<llibreria>
    <llibre>
        <titol>Harry Potter i la pedra filosofal</titol>
        <autor>J. K. Rowling</autor>
        <any>1997</any>
        <preu>25.00</preu>
    </llibre>
    <llibre>
        <titol>El Senyor dels Anells</titol>
        <autor>J. R. R. Tolkien</autor>
        <any>1954</any>
        <preu>32.00</preu>
    </llibre>
    <llibre>
        <titol>Crònica de una mort anunciada</titol>
        <autor>Gabriel Garcia Márquez</autor>
        <any>1981</any>
        <preu>20.00</preu>
    </llibre>
</llibreria>
```

• Nodes

Els documents XML es tracten com arbres de nodes, i hi ha set tipus de nodes en XPath:

1. **Element:** Representa un element com ara **<titol>**.
2. **Atribut:** Com l'atribut **idioma="ca"**.
3. **Text:** Conté el text d'un element com ara **J. K. Rowling**.
4. **Espai de Noms:** Representa l'espai de noms d'un node.
5. **Instrucció de Processament:** Com **<?xml version="1.0" encoding="UTF-8"?>**.
6. **Comentari:** Representa un comentari com **<!-- Aquest és un comentari -->**.
7. **Nodes Arrel:** El node més alt de l'arbre.

• Relacions de Nodes

- **Pare (Parent):** Cada element i atribut té un pare, com ara **<llibre>** és pare de **<titol>**, **<autor>**, **<any>**, i **<preu>**.
- **Fill (Children):** Nodes elements poden tenir zero, un o més fills. Exemple: **<titol>**, **<autor>**, **<any>**, i **<preu>** són fills de **<llibre>**.
- **Germans (Siblings):** Nodes amb el mateix pare. Exemple: **<titol>**, **<autor>**, **<any>**, i **<preu>** són germans.
- **Ancestres (Ancestors):** Pares, avis, etc., d'un node. Exemple: els avantpassats de **<titol>** són **<llibre>** i **<llibreria>**.
- **Descendents (Descendants):** Fills, nets, etc., d'un node. Exemple: els descendents de **<llibreria>** són **<llibre>**, **<titol>**, **<autor>**, **<any>**, i **<preu>**.

• Selecció de Nodes

XPath utilitza expressions de camí per seleccionar nodes o conjunts de nodes en un document XML. A continuació, es mostren algunes expressions útils de camí:

- **titol**: Selecciona tots els nodes amb el nom "titol".
- **/**: Selecciona des de l'element arrel.
- **//**: Selecciona nodes al document des del node actual, independentment de la seva ubicació.
- **.**: Selecciona el node actual.
- **..**: Selecciona el pare del node actual.
- **@**: Selecciona atributs.

• Predicats

Els predicats en XPath són utilitzats per filtrar i seleccionar nodes específics o conjunts de nodes basats en certes condicions o criteris. Aquests predicats s'afegeixen a les expressions de camí i estan incrustats en claudàtors quadrats ([]).

Selecció de Nodes amb Predicats:

/llibreria/llibre[1]: Selecciona el primer **<llibre>** que és fill de l'element **<llibreria>**.
/llibreria/llibre[last()]: Selecciona l'últim **<llibre>** que és fill de l'element **<llibreria>**.
/llibreria/llibre[preu>35.00]: Selecciona tots els **<llibre>** amb un **<preu>** superior a 35.00.

Exemples d'ús de Predicats:

- **Selecció mitjançant Atributs:**

  /llibreria/llibre[@any='1997']: Selecciona tots els llibres amb l'atribut 'any' igual a '1997'.
  /llibreria/llibre[@preu>30.00]: Selecciona tots els llibres amb l'atribut 'preu' superior a 30.00.

- **Selecció basada en Posició:**

  /llibreria/llibre[position()=2]: Selecciona el segon llibre de la biblioteca.
  /llibreria/llibre[position()>1 and position()<4]: Selecciona els llibres que ocupen les posicions 2 i 3.

- **Selecció amb Diferents Condicions:**

  /llibreria/llibre[preu>20.00 and any<2000]: Selecciona els llibres amb un preu superior a 20.00 i un any inferior a 2000.

- **Ús de Funcions:**

  /llibreria/llibre[contains(titol, 'Harry')]: Selecciona tots els llibres amb l'element 'titol' que contingui la paraula 'Harry'.
  → **/llibreria/llibre[starts-with(autor, 'J.')]**: Selecciona tots els llibres amb l'element 'autor' que comenci amb 'J.'.

• Comodins i Seleccions Múltiples

Els comodins de XPath s'utilitzen per seleccionar nodes XML desconeguts. Exemples d'ús de comodins:

- **/llibreria/***: Selecciona tots els nodes fill de l'element llibreria.
- **//***: Selecciona tots els elements al document.
- **//titol[@*]**: Selecciona tots els elements titol que tenen almenys un atribut.

• Eixos XPath

Els eixos representen relacions amb el node de context actual i s'utilitzen per localitzar nodes relatius a aquest node. Exemples d'eixos XPath:

- **ancestor**: Selecciona tots els avantpassats (pare, avi, etc.) del node actual.
- **ancestor-or-self**: Selecciona tots els avantpassats del node actual i el node actual mateix.
- **attribute**: Selecciona tots els atributs del node actual.
- **child**: Selecciona tots els fills del node actual.
- **descendant**: Selecciona tots els descendents (fills, nets, etc.) del node actual.
- **descendant-or-self**: Selecciona tots els descendents del node actual i el node actual mateix.
- **following**: Selecciona tot el que hi ha al document després de la etiqueta de tancament del node actual.
- **following-sibling**: Selecciona tots els germans posteriors al node actual.
- **namespace**: Selecciona tots els nodes d'espai de noms del node actual.
- **parent**: Selecciona el pare del node actual.
- **preceding**: Selecciona tots els nodes que apareixen abans del node actual en el document, excepte els avantpassats, nodes d'atributs i nodes d'espai de noms.
- **preceding-sibling**: Selecciona tots els germans anteriors al node actual.
- **self**: Selecciona el node actual.

• Operadors XPath

Els resultats d'una expressió XPath poden ser un conjunt de nodes, una cadena, un valor booleà o un número. Exemples d'operadors XPath:

- **|**: Calcula dos conjunts de nodes: **//llibre | //cd**
- **+**: Addició: **6 + 4**
- **-**: Resta: **6 - 4**
- ***: Multiplicació: **6 * 4**
- **div**: Divisió: **8 div 4**
- **=**: Igual: **preu=9.80**
- **!=**: No és igual: **preu!=9.80**
- **<**: Menor que: **preu<9.80**
- **<=**: Menor o igual que: **preu<=9.80**
- **>**: Major que: **preu>9.80**
- **>=**: Major o igual que: **preu>=9.80**
- **or**: O: **preu=9.80 or preu=9.70**
- **and**: I: **preu>9.00 and preu<9.90**
- **mod**: Mòdul (resta de la divisió): **5 mod 2**

• Funcions a XPath

- **last() :** Retorna la posició de l'últim node del conjunt actual.
  Exemple: /llibreria/llibre[last()] selecciona l'últim llibre de la biblioteca.

- **position() :** Retorna la posició del node actual en el conjunt.
  Exemple: /llibreria/llibre[position()>1] selecciona els llibres que no són el primer de la biblioteca.
- **count(node-set)**: Retorna el nombre de nodes en el conjunt especificat.
  Exemple: count(/llibreria/llibre) compta el nombre total de llibres a la biblioteca.

- **name():** Retorna el nom del node actual.
  Exemple: /llibreria/llibre/name() retorna "llibre".

- **concat(string1, string2, ...):** Combina diverses cadenes en una sola.
  Exemple: concat(/llibreria/llibre/titol, ' - ', /llibreria/llibre/autor) concatena el títol i l'autor del llibre.

- **contains(string, substring):** Verifica si una cadena conté una altra cadena.
  Exemple: /llibreria/llibre[contains(titol, 'Harry')] selecciona llibres amb la paraula "Harry" al títol.

- **starts-with(string, prefix)::** Verifica si una cadena comença amb una altra cadena.
  Exemple: /llibreria/llibre[starts-with(autor, 'J.')] selecciona llibres amb autors que comencen amb "J.".

- **substring(string, start, length):** Retorna una part de la cadena, començant des de la posició especificada i amb la longitud indicada.
  Exemple: substring(/llibreria/llibre/titol, 1, 3) retorna els primers tres caràcters del títol.

- **sum(node-set)**: Retorna la suma dels valors numèrics del conjunt de nodes especificat.
  Exemple: sum(/llibreria/llibre/preu) calcula la suma dels preus de tots els llibres.

- **normalize-space(string):** Elimina els espais en blanc al principi i al final de la cadena, i redueix els espais interns a un sol espai.
  Exemple: normalize-space(/llibreria/llibre/titol) retorna el títol amb espais en blanc normalitzats.

- **not(expr):** Retorna **true** si l'expressió és **false**, i viceversa.
  Exemple: not(/llibreria/llibre[preu>50.00]) selecciona llibres amb un preu inferior o igual a 50.00.

XSLT (XSL-Transformations)

XSLT és un llenguatge de transformació utilitzat per transformar documents XML en un altre format, com HTML, XHTML o XML diferent.

XSLT:

Permet separar la presentació del contingut

- XSLT és la part més important de XSL
- Transforma un document XML en un altre document XML, HTML, o qualsevol basat en XML
- Utilitza XPath per a navegar pels documents XML
- Recomanació W3C

• Definició XSLT

Un xslt es defineix amb la capçalera:

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

Per associar a un arxiu utilizem:

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

Exemple:

Suposem que volem transformar este **XML** de llibres en una pàgina **HTML** usant **XSLT**

```
<catalog>
    <book>
        <title>Harry Potter i la pedra
filosofal</title>
        <author>J.K. Rowling</author>
        <price>20.00</price>
    </book>
    <book>
        <title>Crim i càstig</title>
        <author>Fyodor Dostoevsky</author>
        <price>18.50</price>
    </book>

    <!-- Més llibres -->

</catalog>
```

```
<html>
<head>
    <title>Llibreria Online</title>
</head>
<body>
<h1>Col·lecció de Llibres</h1>
<div class="book">
    <h2>Harry Potter i la pedra filosofal</h2>
    <p>Autor: J.K. Rowling</p>
    <p>Preu: $20.00</p>
</div>
<div class="book">
    <h2>Crim i càstig</h2>
    <p>Autor: Fyodor Dostoevsky</p>
    <p>Preu: $18.50</p>
</div>
<!-- Més llibres -->

</body>
</html>
```

L'XSLT de transformació seria:

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

    <!-- Plantilla per a transformar l'element 'book' -->
    <xsl:template match="book">

        <div class="book">
            <h2><xsl:value-of select="title"/></h2>
            <p>Autor: <xsl:value-of select="author"/></p>
            <p>Preu: $<xsl:value-of select="price"/></p>
        </div>

    </xsl:template>

    <!-- Plantilla per a transformar l'element 'catalog' -->
    <xsl:template match="catalog">

        <html>
            <head>
                <title>Llibreria Online</title>
            </head>
            <body>
                <h1>Col·lecció de Llibres</h1>
                <xsl:apply-templates select="book"/>
            </body>
        </html>

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

L'arxiu XSLT anterior conté 2 plantilles:

- La primera s'aplica a cada element `'book'` i genera un bloc HTML amb el **títol**, **l'autor** i el **preu** del llibre.
- La segona s'aplica a l'element `'catalog'` i envolta els llibres amb una pàgina **HTML** completa

Es a dir, amb XSLT, cada element `'book'` s'ha convertit en un bloc **HTML** amb el títol, l'autor i el preu del llibre, i **tots** aquests estan continguts en una **pàgina HTML** completa.

• Plantilles (Templates):

Les plantilles són la base de les transformacions XSLT, una plantilla especifica com s'ha de transformar un element o grup d'elements d'un document XML. Cada plantilla pot contindre regles que indiquen com processar diferents parts del document.

Les plantilles les definim amb l'element <**xsl:template**>, que conté un atribut **match** que especifica el patró d'elements que s'aplicarà a la plantilla.

Exemple:

```
<!-- Plantilla per a transformar l'element 'book' -->
<xsl:template match="book">

    <!-- Contingut de la plantilla -->

</xsl:template>
```

A l'exemple, la plantilla s'aplicarà a cada element **'book'** del nostre document XML.

Per desenvolupar una plantilla, afegirem **contingut específic dins del bloc de la plantilla**. Per exemple, si volem transformar l'element **'book'** en un bloc **HTML**, la plantilla podria ser:

```
<!-- Plantilla per a transformar l'element 'book' -->
<xsl:template match="book">
    <div class="book">
        <h2><xsl:value-of select="title"/></h2>
        <p>Autor: <xsl:value-of select="author"/></p>
        <p>Preu: $<xsl:value-of select="price"/></p>
    </div>
</xsl:template>
```

Cada element **'book'** serà transformar en un bloc <**div**> amb l'estructura HTML especificada.

Aplicació de les plantilles.

Una volta hem definit les nostres plantilles, les podem aplicar a diferents nodes del nostre document **XML** utilitzant l'element **<xsl:apply-templates>**.

<xsl:apply-templates> indica quina plantilla s'ha d'utilitzar per a cada tipus d'element. Per exemple:

```
<!-- Aplica la plantilla als nodes 'book' dins de 'catalog' -->
<xsl:apply-templates select="catalog/book"/>
```

Amb aquesta línia, apliquem la plantilla als nodes **'book'** continguts dins de l'element **'catalog'**.

- ## Selecció de Nodes:

La selecció de nodes en XSLT és una part fonamental del procés de transformació. Ens permet apuntar als elements específics del nostre document XML als quals volem aplicar les plantilles.

Utilitzem <**xsl:apply-templates**> per seleccionar i aplicar plantilles a nodes específics. Pot contidre un atribut **select**.

Select: Especifica la ruta **d'XPath** per seleccionar els nodes desitjats.

```
<!-- Aplica la plantilla als nodes 'book' dins de 'catalog' -->
<xsl:apply-templates select="catalog/book"/>
```

Selecció de Nodes amb XPath:

XPath és una part integral de XSLT, ja que s'utilitza per navegar pels documents XML i seleccionar nodes específics. Amb XPath, podem utilitzar patrons i expressions per afinar la nostra selecció.

```
<!-- Aplica la plantilla als nodes 'autor' dins de 'llibre' amb preu superior a 50 -->
<xsl:apply-templates select="llibre[preu > 50]/autor"/>
```

A l'exemple, apliquem la plantilla als nodes **'autor'** continguts dins de l'element **'llibre'** només si el preu és **superior a 50**.

Exemple:

```
<!-- Plantilla per a transformar l'element 'catalog' -->
<xsl:template match="catalog">
  <html>
    <head>
      <title>Col·lecció de Llibres</title>
    </head>
    <body>
      <h1>Llibreria Online</h1>
      <!-- Aplica la plantilla als nodes 'book' dins de 'catalog' -->
      <xsl:apply-templates select="book"/>
    </body>
  </html>
</xsl:template>
```

Apliquem la plantilla als nodes **'book'** continguts dins de l'element **'catalog'**.

Amb aquesta transformació, el document **XML** inicial es convertirà en un document **HTML** amb l'estructura especificada a les plantilles.

• Valor de Nodes:

L'element **<xsl:value-of>** s'utilitza per recuperar i mostrar el contingut d'un node XML. Aquest element s'utilitza dins de les plantilles **per extreure la informació desitjada i incrustar-la** en el document de sortida.

```
<!-- Mostra el títol del llibre -->
<xsl:value-of select="title"/>
```

A l'exemple, estem extreient i mostrant el contingut del node **'title'** dins de l'element **'book'**.

L'element **<xsl:value-of>** normalment s'utilitza dins de les plantilles per mostrar informació específica dels nodes.
Per exemple, si volem millorar la nostra plantilla per incloure el **títol** , **l'autor**, i el **preu** d'un llibre dins del nostre document HTML:

```
<!-- Plantilla per a transformar l'element 'book' -->
<xsl:template match="book">
  <div class="book">
    <h2><xsl:value-of select="title"/></h2>
    <p>Autor: <xsl:value-of select="author"/></p>
    <p>Preu: $<xsl:value-of select="price"/></p>
  </div>
</xsl:template>
```

• Estructures Condicionals amb <xsl:if> , < xsl:choose>

L'element **<xsl:if>** s'utilitza per aplicar condicions als nodes del nostre document XML.

Per exemple, si volem condicionar la visualització d'una etiqueta basada en el preu del llibre. Si el preu és superior a 50, afegim la classe "**car**" (€ de "caro")

```
<!-- Si 'preu' és major de 50, aplica aquesta plantilla -->
<xsl:if test="price > 50">
  <span class="car">Car</span>
</xsl:if>
```

Amb este codi, estem comprovant si el preu del llibre és superior a 50. Si la condició és certa, afegim la classe "car" al nostre document HTML.

L'element **<xsl:choose>** proporciona una estructura més avançada per gestionar múltiples condicions. Aquest element s'utilitza quan hi ha diverses opcions i només volem executar la primera condició que es compleix

```
<!-- Estructura amb xsl:choose -->
<xsl:choose>
    <!-- Primera condició -->
    <xsl:when test="preu > 50">
        <span class="car">Car</span>
    </xsl:when>
    <!-- Segona condició -->
    <xsl:when test="preu &gt; 25 and preu &lt;= 50">
        <span class="mitja">Mitjà</span>
    </xsl:when>
    <!-- Condició per defecte -->
    <xsl:otherwise>
        <span class="economic">Econòmic</span>
    </xsl:otherwise>
</xsl:choose>
```

A l'exemple, **<xsl:choose>** inclou múltiples **<xsl:when>** per a les diferents condicions, i **<xsl:otherwise>** per a la condició per defecte. La primera condició que es compleix s'executarà.

- ## Iteració amb <xsl:for-each>

L'element **<xsl:for-each>** permet recórrer múltiples nodes del teu document **XML** i aplicar plantilles a cadascun d'ells.
Per exemple, podem utilitzar **<xsl:for-each>** per recórrer tots els nodes **'book'** i aplicar la plantilla corresponent:

```
<!-- Recorre tots els nodes 'book' i aplica la plantilla -->
<xsl:for-each select="catalog/book">
  <!-- Contingut de la plantilla -->
</xsl:for-each>
```

Amb este codi, estem recorrent tots els nodes **'book'** dins de l'element **'catalog'** i aplicant la plantilla definida.

- ## Creació i Ús de Variables

Les variables en **XSLT** són útils per **emmagatzemar valors temporals** que es poden utilitzar en diferents parts de les nostres transformacions. s defineixen ambl'element **<xsl:variable>.**

<xsl:variable> permet assignar un valor a una variable i fer-ne referència en altres parts del codi. Per exemple:

```
<!-- Defineix una variable amb el nom 'color' -->
<xsl:variable name="color" select="'#f2f2f2'"/>
```

En este exemple, hem creat una variable anomenada **'color'** amb el valor literal **'#f2f2f2'**. Podem canviar el valor de la variable segons les nostres necessitats.

Constants

Les variables poden ser utilitzades com a **constants**, permetent definir valors reutilitzables. Per exemple, si volem utilitzar una variable per representar un descompte per a tots els llibres:

```
<!-- Defineix una constant amb el nom 'descompte' -->
<xsl:variable name="descompteFixe" select="0.1"/>
```

`descompteFixe` conté el valor decimal 0.1 que podríem utilitzar com a descompte fixe en tota la transformació.

Pas de Paràmetres a les Plantilles

Les variables també poden ser utilitzades per passar paràmetres a les plantilles, permetent una major flexibilitat en la reutilització de plantilles amb diferents valors. En este cas, volem passar el descompte com a paràmetre:

```
<!-- Exemple de plantilla que rep un paràmetre de descompte -->
<xsl:template match="book">
    <div style="background-color:{$color};">
        <h2><xsl:value-of select="title"/></h2>
        <p>Autor: <xsl:value-of select="author"/></p>
        <p>Preu amb Descompte:
                    $<xsl:value-of select="price - (price * $descompteFixe)"/></p>

    </div>
</xsl:template>
```

En aquest exemple, estem passant el valor de la variable 'descompte' com a paràmetre a una plantilla que pot utilitzar-lo durant la transformació.

Ús de $ en XSLT

La notació **$** s'utilitza per fer referència a una variable en XSLT. Quan volem utilitzar el contingut d'una variable, simplement afegim **$** seguit pel nom de la variable.

```
<!-- Exemple d'ús de la variable 'colorFons' -->
<div style="background-color:{$color};">

<!-- Contingut de la plantilla -->
</div>
```

A XSLT, dins de les cadenes de text, les claus `{ }` indiquen que hi ha una expressió que s'avaluarà i es substituirà pel seu valor. Es a dir, les claus són utilitzades per a incrustar expressions XPath dins de les cadenes de text.

A l'exemple, `style="background-color:{$color};"` indica que el valor de la variable `color` s'avaluarà i s'incrustarà a la posició corresponent en el text. És una manera d'inserir dinàmicament contingut en funció del valor de variables o expressions.

Les plantilles en XSLT poden rebre paràmetres per fer-les més flexibles i reutilitzables. Això permet passar valors específics a una plantilla quan és aplicada.

```
<!-- Definició d'una variable com a paràmetre -->
<xsl:param name="descompte" />
```

Amb esta línia, definim una variable anomenada " **descompte** " com a paràmetre, que serà utilitzat per passar un valor específic quan apliquem la plantilla.

```
 <!-- Plantilla que rep el paràmetre de descompte -->
<xsl:template match="book">
<div style="background-color:{$color};">
    <h2><xsl:value-of select="title"/></h2>
    <p>Autor: <xsl:value-of select="author"/></p>

    <!-- Utilització del paràmetre a la plantilla -->
    <p>Preu amb Descompte: $<xsl:value-of select="price - (price * $descompte)"/></p>
</div>
</xsl:template>
```

A l'anterior plantilla, utilitzem el paràmetre **$descompte** com si fos una variable normal. Este paràmetre permet ajustar el comportament de la plantilla en funció del valor que se li passe.

```
<!-- Exemple d'ús de la plantilla amb el paràmetre -->
<xsl:apply-templates select="catalog/book">
<!-- Passem el valor del descompte com a paràmetre -->
<xsl:with-param name="descompte" select="0.1"/>
</xsl:apply-templates>
```

En este exemple d'ús, estem aplicant la plantilla als nodes "**book**" continguts dins de l'element "**catalog**". A més, mitjançant **<xsl:with-param>**, estem passant un valor específic (0.1) pel paràmetre descompte. Aquest valor pot variar per adaptar-se a diferents situacions o necessitats de transformació.

En resum

Amb **<xsl:param name="descompte" />,** definim un paràmetre anomenat **descompte**.
A la plantilla **<xsl:template match="book">**, podem utilitzar el paràmetre **$descompte** com qualsevol altra variable.
A **<xsl:apply-templates>,** amb <**xsl:with-param**>, passem un valor específic pel paràmetre **descompte**.

- **Exemple Complet d'XSLT:**

Exemple d'arxiu XSLT que pot transformar un document XML a HTML.

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

    <!-- Definició d'una variable -->
    <xsl:variable name="colorFons" select="'#f2f2f2'"/>

    <!-- Plantilla per a transformar l'element 'book' -->
    <xsl:template match="book">
        <div style="background-color:{$colorFons};">
            <h2><xsl:value-of select="title"/></h2>
            <p>Autor: <xsl:value-of select="author"/></p>
            <p>Preu: $<xsl:value-of select="price"/></p>
        </div>
    </xsl:template>

    <!-- Plantilla per a transformar l'element 'catalog' -->
    <xsl:template match="catalog">
        <html>
            <head>
                <title>Llibreria Online</title>
            </head>
            <body>
                <h1>Col·lecció de Llibres</h1>
                <!-- Aplica la plantilla als nodes 'book' dins de 'catalog' -->
                <xsl:apply-templates select="book"/>
            </body>
        </html>
    </xsl:template>

</xsl:stylesheet>
```