Trabajo XPATH XSLT XQuery

José María Fernández Saavedra

1º DAW

Introducción a XPath

XPath (XML Path Language) es un lenguaje de consulta diseñado para seleccionar nodos específicos dentro de un documento XML. Su importancia radica en que permite navegar por la estructura jerárquica de un XML de forma precisa, extrayendo elementos, atributos o texto según las necesidades. Es una herramienta clave en tecnologías como XSLT (para transformar XML) y XQuery (para consultar XML), ya que facilita el acceso a datos estructurados.

Por ejemplo, en un documento XML que representa una librería, XPath puede usarse para seleccionar el título de un libro específico o todos los libros de un género determinado. Su simplicidad y flexibilidad lo hacen esencial para procesar XML en aplicaciones web y sistemas de gestión de datos.

Cómo se Utiliza XPath

XPath funciona como una "ruta" que recorre la estructura de un XML, similar a cómo se navega por carpetas en un sistema de archivos. Cada elemento, atributo o texto en el XML es un nodo, y XPath usa expresiones para localizarlos. Por ejemplo:

- Para seleccionar el título del primer libro en un XML de una librería: /bookstore/book[1]/title.
- Para obtener todos los autores: /bookstore/book/author.

Estas expresiones se usan en XSLT para transformar datos o en XQuery para realizar consultas, permitiendo extraer información específica sin necesidad de recorrer manualmente el documento.

Sintaxis de XPath

La sintaxis de XPath se basa en expresiones que especifican rutas hacia los nodos. Hay dos tipos principales de rutas:

- **Rutas absolutas**: Comienzan desde la raíz del documento con /. Ejemplo: /bookstore/book/title selecciona todos los títulos de los libros dentro de <bookstore>.
- **Rutas relativas**: No comienzan desde la raíz, sino desde un contexto actual. Ejemplo: book/title selecciona los títulos de los libros en el contexto actual.

Además, XPath permite seleccionar:

- **Elementos**: book selecciona todos los nodos <book>.
- **Atributos**: @id selecciona el atributo id de un nodo. Ejemplo: /bookstore/book/@id devuelve los IDs de todos los libros.
- **Texto**: text() extrae el contenido textual de un nodo.

Navegación de Nodos

XPath navega por la estructura jerárquica del XML usando ejes como hijos, padres o hermanos. Por ejemplo, en el XML:

```xml
<bookstore>
  <book id="1">
    <title>El Quijote</title>
    <author>Miguel de Cervantes</author>
    <price>19.99</price>
    <genre>Clásico</genre>
  </book>
  <!-- Más libros -->
</bookstore>
```

Algunas expresiones XPath para navegar serían:

- /bookstore/book[1]/title: Selecciona el título del primer libro ("El Quijote").
- /bookstore/book/author: Selecciona todos los autores de los libros.
- /bookstore/book[@id="2"]/price: Selecciona el precio del libro con id="2".

Estas expresiones permiten acceder a nodos específicos en cualquier nivel del XML.

Predicados

Los predicados en XPath se usan para filtrar nodos según condiciones, y se escriben entre corchetes [ ]. Son esenciales para selecciones precisas. Ejemplos:

- /bookstore/book[price > 20]: Selecciona los libros cuyo precio es mayor a 20.
- /bookstore/book[genre = 'Ficción']/title: Selecciona los títulos de los libros de género "Ficción".
- /bookstore/book[position() = 1]: Selecciona el primer libro.

Por ejemplo, para obtener los títulos de los libros de Gabriel García Márquez:

```
/bookstore/book[author = 'Gabriel García Márquez']/title
```

Esto devuelve "Cien años de soledad" y "Crónica de una muerte anunciada" (si están en el XML).

Funciones XPath

XPath incluye funciones integradas para operaciones avanzadas. Algunas comunes son:

- **count()**: Cuenta nodos. Ejemplo: count(/bookstore/book) devuelve el número total de libros.
- **concat()**: Une cadenas. Ejemplo: concat(title, ' por ', author) genera "El Quijote por Miguel de Cervantes".
- **substring()**: Extrae partes de una cadena. Ejemplo: substring(title, 1, 5) toma los primeros 5 caracteres del título.
- **string()**: Convierte un nodo a texto. Ejemplo: string(/bookstore/book[1]/price) devuelve "19.99" como texto.

Estas funciones se combinan con rutas y predicados para realizar selecciones complejas, como contar los libros de un género (count(/bookstore/book[genre = 'Clásico'])).

Introducción a XSLT

XSLT (Extensible Stylesheet Language Transformations) es un lenguaje diseñado para transformar documentos XML en otros formatos, como HTML, XML o texto plano. Su propósito es definir reglas que convierten la estructura y contenido de un XML en una presentación diferente, ya sea para mostrar datos de forma visualmente atractiva (por ejemplo, en una página web) o para adaptarlos a otro sistema. XSLT es crucial en el procesamiento de XML porque permite separar los datos (XML) de su presentación, facilitando la reutilización y personalización.

Por ejemplo, un XML con datos de una librería puede transformarse en una tabla HTML para un sitio web o en un documento PDF con un formato específico. XSLT usa expresiones XPath para seleccionar datos del XML, lo que lo hace poderoso y flexible en aplicaciones web y sistemas de gestión de información.

Uso de XSLT

XSLT funciona mediante hojas de estilo que contienen reglas de transformación. Cada regla especifica cómo un nodo o grupo de nodos del XML debe convertirse en el formato de salida. Por ejemplo:

- Para mostrar una lista de libros en HTML, XSLT recorre los nodos <book> y genera etiquetas <tr> y <td> para una tabla.
- Para cambiar el formato, XSLT puede renombrar elementos o agregar estilos visuales.

Estas transformaciones se aplican con herramientas como procesadores XSLT (ej., xsltproc) o navegadores web, produciendo resultados como páginas web estilizadas o nuevos documentos XML.

Sintaxis y Estructura Básica de XSLT

Una hoja de estilo XSLT es un documento XML que comienza con el elemento <xsl:stylesheet> y usa el espacio de nombres xmlns:xsl="http://www.w3.org/1999/XSL/Transform". Su estructura básica incluye plantillas (<xsl:template>) que definen cómo transformar partes del XML. Los elementos clave son:

- **<xsl:stylesheet>**: Define la hoja de estilo y su versión (ej., version="1.0").
- **<xsl:template>**: Especifica una regla de transformación para un nodo. El atributo match indica qué nodos procesar (ej., match="/", match="book").
- **<xsl:apply-templates>**: Aplica plantillas a los nodos hijos, permitiendo procesar el XML recursivamente.
- **<xsl:value-of>**: Extrae el valor de un nodo. Ejemplo: <xsl:value-of select="title"/> muestra el título de un libro.
- **<xsl:for-each>**: Itera sobre una lista de nodos. Ejemplo: <xsl:for-each select="/bookstore/book"> recorre todos los libros.
- **<xsl:if>**: Aplica una condición. Ejemplo: <xsl:if test="price > 20"> procesa solo libros caros.
- **<xsl:choose>, <xsl:when>, <xsl:otherwise>**: Permite decisiones condicionales. Ejemplo: Cambiar el color de fondo según el género.
- **<xsl:element>**: Crea un elemento en la salida. Ejemplo: <xsl:element name="div"> genera .
- **<xsl:attribute>**: Añade atributos a un elemento. Ejemplo: <xsl:attribute name="class">book</xsl:attribute>.

Ejemplo de Hoja de Estilo XSLT

A continuación, se muestra un fragmento de una hoja de estilo XSLT que transforma el XML de una librería (books.xml) en una tabla HTML, usando los elementos mencionados:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head><title>Librería</title></head>
      <body>
        <h1>Catálogo de Libros</h1>
        <table border="1">
          <tr><th>Título</th><th>Autor</th><th>Precio
(€)</th><th>Género</th></tr>
          <xsl:apply-templates select="/bookstore/book"/>
        </table>
      </body>
    </html>
  </xsl:template>
```

```
  <xsl:template match="book">
    <tr>
      <xsl:element name="td">
        <xsl:value-of select="title"/>
      </xsl:element>
      <xsl:element name="td">
        <xsl:value-of select="author"/>
      </xsl:element>
      <xsl:element name="td">
        <xsl:if test="price > 20">
          <xsl:attribute name="style">color: red;</xsl:attribute>
        </xsl:if>
        <xsl:value-of select="price"/>
      </xsl:element>
      <xsl:element name="td">
        <xsl:choose>
          <xsl:when test="genre = 'Clásico'">
            <xsl:attribute name="style">background-color:
lightblue;</xsl:attribute>
          </xsl:when>
          <xsl:otherwise>
            <xsl:attribute name="style">background-color:
white;</xsl:attribute>
          </xsl:otherwise>
        </xsl:choose>
        <xsl:value-of select="genre"/>
      </xsl:element>
    </tr>
  </xsl:template>
</xsl:stylesheet>
```

Explicación:

- **<xsl:template match="/">**: Procesa la raíz del XML, generando la estructura HTML.
- **<xsl:apply-templates>**: Aplica la plantilla para cada <book>.
- **<xsl:for-each>**: No se usa aquí, pero podría reemplazar <xsl:apply-templates> para iterar directamente.
- **<xsl:value-of>**: Muestra el título, autor, precio y género.
- **<xsl:if>**: Colorea en rojo los precios mayores a 20.
- **<xsl:choose>**: Aplica fondo azul a los géneros "Clásico".
- **<xsl:element> y <xsl:attribute>**: Crean celdas <td> con estilos dinámicos.

Este XSLT produce una tabla HTML estilizada, mostrando cómo los elementos trabajan juntos para transformar el XML de forma condicional y estructurada.

Introducción a XQuery

XQuery es un lenguaje de consulta diseñado para procesar y manipular datos almacenados en documentos XML. Su propósito es extraer información específica, transformar datos y generar nuevos formatos, de manera similar a SQL para bases de datos relacionales. XQuery es importante porque permite realizar consultas complejas sobre datos estructurados en XML, lo que lo hace esencial en aplicaciones que manejan formatos como RSS, configuraciones o intercambio de datos.

Por ejemplo, en un XML de una librería, XQuery puede listar todos los libros con un precio superior a 20 euros o generar un informe con los títulos ordenados alfabéticamente. Su capacidad para combinar consultas y transformaciones lo convierte en una herramienta poderosa para desarrolladores web y gestores de información.

Uso de XQuery

XQuery se utiliza para:

- **Consultar**: Extraer datos específicos, como los autores de libros de un género.
- **Transformar**: Convertir XML en otros formatos, como HTML o texto.
- **Manipular**: Modificar o combinar datos, por ejemplo, calcular el precio promedio de los libros.

XQuery emplea expresiones XPath para navegar por el XML y estructuras como FLWOR para consultas avanzadas. Se ejecuta con herramientas como BaseX, Saxon o procesadores integrados en aplicaciones.

Sintaxis de XQuery: Construcción FLWOR

La construcción FLWOR (For, Let, Where, Order by, Return) es la base de las consultas en XQuery, permitiendo procesar datos de forma estructurada y eficiente. Cada componente tiene un propósito específico:

- **For**: Itera sobre una secuencia de nodos. Ejemplo: for $book in doc("books.xml")/bookstore/book recorre todos los libros.
- **Let**: Asigna valores a variables. Ejemplo: let $max_price := 30 define un precio máximo.
- **Where**: Filtra los nodos según una condición. Ejemplo: where $book/price > 20 selecciona libros caros.
- **Order by**: Ordena los resultados. Ejemplo: order by $book/title ordena por título alfabéticamente.
- **Return**: Especifica qué devolver. Ejemplo: return $book/title devuelve los títulos.

Ejemplo de Consulta FLWOR

Usando el XML books.xml:

```
<bookstore>
  <book id="1"><title>El Quijote</title><author>Miguel de
Cervantes</author><price>19.99</price><genre>Clásico</genre></book>
  <book id="2"><title>Cien años de soledad</title><author>Gabriel García
Márquez</author><price>25.50</price><genre>Ficción</genre></book>
  <!-- Más libros -->
</bookstore>
```

Una consulta XQuery para listar los títulos y precios de los libros con precio mayor a 20, ordenados por título, sería:

```
for $book in doc("books.xml")/bookstore/book
let $currency := "€"
where $book/price > 20
order by $book/title
return <result>
  <title>{$book/title/text()}</title>
  <price>{$book/price/text() || $currency}</price>
</result>
```

Explicación:

- **For**: Recorre cada <book> en el XML.
- **Let**: Define la variable $currency para añadir "€" al precio.
- **Where**: Filtra libros con price > 20.
- **Order by**: Ordena los resultados por el título.
- **Return**: Devuelve un elemento <result> con el título y el precio formateado (ej., "Cien años de soledad, 25.50€").

Esta consulta produce una salida XML con los libros que cumplen la condición, demostrando la potencia de FLWOR para consultas estructuradas.

Ejemplo de Uso

En esta sección, se presenta un ejemplo práctico que demuestra el uso de XPath, XSLT y XQuery con un documento XML. El ejemplo se basa en un catálogo de ratones gaming, representado en el archivo gaming_mice.xml, que se transforma en HTML mediante XSLT y se consulta con XQuery.

XML: gaming_mice.xml

El archivo gaming_mice.xml contiene un catálogo de 20 ratones gaming, con un total de aproximadamente 104 líneas. Cada ratón está representado por un elemento <mouse> que incluye:

- <brand>: Marca del ratón (ej., Logitech, Razer).
- <model>: Modelo específico (ej., G Pro X Superlight).
- <price>: Precio en euros (entre 59.99 y 159.99).
- <dpi>: Sensibilidad máxima (entre 3200 y 36000).
- <connection>: Tipo de conexión (Wired o Wireless).
- <weight>: Peso en gramos (entre 58 y 142).

Ejemplo de un nodo:

```
<mouse>
  <brand>Logitech</brand>
  <model>G Pro X Superlight</model>
  <price>149.99</price>
  <dpi>25600</dpi>
  <connection>Wireless</connection>
  <weight>63</weight>
</mouse>
```

El XML usa XPath en las transformaciones y consultas, como /mice_catalog/mouse para seleccionar todos los ratones.

XSLT: gaming_mice.xsl

El archivo gaming_mice.xsl transforma gaming_mice.xml en una tabla HTML estilizada (gaming_mice.html). La hoja de estilo utiliza elementos XSLT como <xsl:stylesheet>, <xsl:template>, <xsl:for-each>, <xsl:value-of>, <xsl:choose>, <xsl:if>, <xsl:element> y <xsl:attribute> para generar una tabla con las siguientes características:

- **Columnas**: Marca, modelo, precio, DPI, conexión y peso.
- **Estilos**:
    - Precios superiores a 100€ en rojo, inferiores en verde.
    - Conexiones inalámbricas en negrita.
    - Pesos envueltos en un elemento con clase.
    - Filas alternas con fondo gris y efecto hover.

Fragmento clave del XSLT:

```
<xsl:for-each select="mice_catalog/mouse">
  <tr>
    <td><xsl:value-of select="brand"/></td>
    <td><xsl:value-of select="model"/></td>
    <td>
      <xsl:choose>
        <xsl:when test="price > 100">
          <xsl:attribute name="style">color: red;</xsl:attribute>
        </xsl:when>
        <xsl:otherwise>
          <xsl:attribute name="style">color: green;</xsl:attribute>
        </xsl:otherwise>
      </xsl:choose>
      <xsl:value-of select="price"/>
    </td>
    <td><xsl:value-of select="dpi"/></td>
    <td>
      <xsl:if test="connection = 'Wireless'">
        <xsl:attribute name="style">font-weight: bold;</xsl:attribute>
      </xsl:if>
      <xsl:value-of select="connection"/>
    </td>
    <td><xsl:element name="span"><xsl:attribute
name="class">weight</xsl:attribute><xsl:value-of
select="weight"/></xsl:element></td>
  </tr>
</xsl:for-each>
```

La transformación se realizó usando una herramienta online (https://www.freeformatter.com/xsl-transformer.html), generando gaming_mice.html. El resultado es una tabla visualmente atractiva, con estilos condicionales basados en los datos del XML.

Catálogo de Ratones Gaming

Marca	Modelo	Precio (€)	DPI	Conexión	Peso (g)
Logitech	G Pro X Superlight	149.99	25600	**Wireless**	63
Razer	DeathAdder V3 Pro	159.99	30000	**Wireless**	63
SteelSeries	Aerox 5 Wireless	139.99	18000	**Wireless**	74
HyperX	Pulsefire Haste	59.99	16000	Wired	59
Corsair	Dark Core RGB Pro	89.99	18000	**Wireless**	142
Roccat	Kone Pro Air	129.99	19000	**Wireless**	75

XQuery: query.xq

Se creó una consulta XQuery (query.xq) para extraer los modelos y precios de los ratones con un precio superior a 100€, ordenados por modelo. La consulta usa la construcción FLWOR y expresiones XPath:

```
for $mouse in doc("gaming_mice.xml")/mice_catalog/mouse
where $mouse/price > 100
order by $mouse/model
return <result>
  <model>{$mouse/model/text()}</model>
  <price>{$mouse/price/text()}€</price>
</result>
```

Explicación:

- **For**: Itera sobre cada <mouse> en /mice_catalog/mouse.
- **Where**: Filtra ratones con price > 100.
- **Order by**: Ordena los resultados por el campo <model>.
- **Return**: Devuelve un elemento <result> con el modelo y el precio, añadiendo "€".

La consulta se probó con BaseX, produciendo una salida XML con los ratones que cumplen la condición (ej., "G Pro X Superlight, 149.99€"). Esto demuestra cómo XQuery usa XPath (/mice_catalog/mouse) y FLWOR para consultas precisas.

Webs Consultadas

- W3C XPath: https://www.w3.org/TR/xpath/
- W3C XSLT: https://www.w3.org/TR/xslt/
- W3C XQuery: https://www.w3.org/TR/xquery-31/
- W3Schools XPath: https://www.w3schools.com/xml/xpath_intro.asp
- W3Schools XSLT: https://www.w3schools.com/xml/xsl_intro.asp
- W3Schools XQuery: https://www.w3schools.com/xml/xquery_intro.asp
- Grok 3, xAI: Asistencia en redacción y ejemplos prácticos.
- ChatGPT: Asistencia en redacción y ejemplos prácticos.