

Ejercicios de XML resueltos

Los 2 ejercicios de XML de la web en una sola hoja: cada uno con su enunciado, los datos que necesitas (código de partida, ejemplos de entrada y salida o la base de datos) y, al final, la solución explicada.

Índice

Ejercicios largos

1. Catálogo de una editorial: XPath, XQuery, XSLT y XSD

2. Observaciones meteorológicas: XPath, XQuery, XSLT a CSV y DTD

Ejercicios largos

1. Catálogo de una editorial: XPath, XQuery, XSLT y XSD

Difícil · 90 minutos · apuntesdam.com/ejercicios/xml/catalogo-de-una-editorial-xml

Una editorial pequeña publica su catálogo en XML para que las librerías lo carguen en sus programas. El documento tiene un elemento `libro` por título, con el código, el género y los ejemplares en stock como atributos, y el título, los autores (uno o varios), el año, el precio y las páginas como elementos.

Con ese único documento vas a hacer lo que se hace en el día a día con XML: buscar datos con XPath, sacar informes y construir XML nuevo con XQuery, presentarlo como página web con XSLT y, por último, escribir el XML Schema que garantiza que el documento que reciben las librerías es correcto.

Los apartados se pueden hacer en cualquier orden, pero conviene seguirlo: cada herramienta se apoya en la anterior (XQuery y XSLT usan expresiones XPath, y el XSD formaliza lo que has aprendido de la estructura).

Requisitos

- Resuelve cada apartado con la herramienta que indica, sobre el documento `catalogo.xml`.
- Cada apartado se corrige por separado. Las consultas XPath y XQuery se comparan con el resultado de la solución sobre el documento (importan los valores y su orden); las transformaciones XSLT, con la salida esperada (sin contar espacios y saltos de línea); y los esquemas, con documentos que deben aceptar y otros que deben rechazar.
- Cuando un apartado pide «el texto» de algo, la consulta debe devolver cadenas (con `text()` o `string()`), no elementos.

```
<?xml version="1.0" encoding="UTF-8"?>
<catalogo editorial="Ediciones del Sur" actualizado="2026-09-30">
  <libro codigo="L001" genero="novela" stock="12">
    <titulo>La casa de las palabras</titulo>
    <autor>Marta Ruiz</autor>
    <anio>2019</anio>
    <precio>18.90</precio>
    <paginas>320</paginas>
  </libro>
  <libro codigo="L002" genero="poesia" stock="0">
    <titulo>Mar adentro, versos</titulo>
    <autor>Julio Peña</autor>
    <anio>2021</anio>
    <precio>12.50</precio>
    <paginas>96</paginas>
  </libro>
  <libro codigo="L003" genero="ensayo" stock="5">
    <titulo>Breve historia de la imprenta</titulo>
    <autor>Ana Gil</autor>
    <autor>Luis Mora</autor>
    <anio>2017</anio>
    <precio>24.00</precio>
    <paginas>410</paginas>
  </libro>
  <libro codigo="L004" genero="novela" stock="3">
    <titulo>El último faro</titulo>
    <autor>Marta Ruiz</autor>
    <anio>2023</anio>
    <precio>21.50</precio>
    <paginas>288</paginas>
  </libro>
  <libro codigo="L005" genero="infantil" stock="25">
    <titulo>Lola y el dragón azul</titulo>
    <autor>Pablo Sanz</autor>
    <anio>2022</anio>
    <precio>9.95</precio>
    <paginas>48</paginas>
  </libro>
  <libro codigo="L006" genero="novela" stock="0">
    <titulo>Ciudades de niebla</titulo>
    <autor>Elena Ríos</autor>
    <anio>2015</anio>
    <precio>19.90</precio>
    <paginas>352</paginas>
  </libro>
  <libro codigo="L007" genero="ensayo" stock="8">
    <titulo>Pensar con datos</titulo>
    <autor>Luis Mora</autor>
    <anio>2024</anio>
    <precio>27.50</precio>
    <paginas>264</paginas>
  </libro>
  <libro codigo="L008" genero="infantil" stock="14">
    <titulo>El bosque que cantaba</titulo>
    <autor>Sara Molina</autor>
    <autor>Pablo Sanz</autor>
    <anio>2020</anio>
    <precio>11.00</precio>
    <paginas>40</paginas>
  </libro>
</catalogo>
```

Apartados

- El texto de los títulos de las novelas, en el orden del documento.
- El texto del título del libro más caro.
- Cuántos libros no tienen ningún ejemplar en stock.
- El texto de los títulos de los libros escritos por más de un autor, en el orden del documento.

- Un informe por género, en orden alfabético: una cadena por género con el formato `género (N): precio medio X €`, donde N es el número de libros y X el precio medio redondeado a dos decimales (sin ceros sobrantes: `25.75`, `20.1`).
- La oferta del mes: para cada libro con más de 10 ejemplares, construye `<oferta codigo="..."><titulo>...</titulo><precio>...</precio></oferta>` con su código, su título y su precio rebajado un 20 % redondeado a dos decimales, ordenadas de menor a mayor precio rebajado.
- Transforma el catálogo en una tabla HTML: `<table>` con la fila de cabecera `<tr><th>Título</th><th>Autores</th><th>Precio</th></tr>` y una fila por libro, ordenadas por título, con los autores separados por coma y espacio, el precio con dos decimales seguido de un espacio y € (`18.90 €`), y el atributo `class="agotado"` en las filas de los libros sin stock.
- Escribe el XML Schema del catálogo: `catalogo` con los atributos obligatorios `editorial` (texto) y `actualizado` (fecha) y uno o más `libro`; cada libro con los atributos obligatorios `codigo` (una L y tres cifras), `genero` (novela, poesía, ensayo o infantil) y `stock` (entero no negativo), y los elementos, en este orden, `titulo`, uno o más `autor`, `anio` (entero de 1450 a 2026), `precio` (decimal no negativo con dos decimales como mucho) y `paginas` (entero positivo).

2. Observaciones meteorológicas: XPath, XQuery, XSLT a CSV y DTD

Difícil · 80 minutos · apuntesdam.com/ejercicios/xml/observaciones-meteorologicas-xml

Varios institutos tienen una estación meteorológica en el tejado y comparten sus datos en un documento XML común: una `estacion` por instituto, con sus atributos (identificador, nombre, provincia y altitud), y dentro un `dia` por cada día con la temperatura máxima, la mínima y la lluvia en milímetros.

El documento declara su DTD en el `DOCTYPE`, pero esa DTD todavía no existe: el último apartado es escribirla. Antes, vas a responder con XPath y XQuery a las preguntas que se hace cualquiera con estos datos (dónde ha llovido más, qué estación es más fría) y a exportarlos en CSV con XSLT para abrirlos en una hoja de cálculo.

Requisitos

- Resuelve cada apartado con la herramienta que indica, sobre el documento `observaciones.xml`.
- Cada apartado se corrige por separado. Las consultas XPath y XQuery se comparan con el resultado de la solución sobre el documento (importan los valores y su orden); las transformaciones XSLT, con la salida esperada (sin contar espacios y saltos de línea); y los esquemas, con documentos que deben aceptar y otros que deben rechazar.
- Cuando un apartado pide «el texto» de algo, la consulta debe devolver cadenas (con `text()` o `string()`), no elementos.

observaciones.xml (xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE observaciones SYSTEM "observaciones.dtd">
<observaciones red="Estaciones de los institutos" periodo="2026-09">
  <estacion id="E01" nombre="Madrid Retiro" provincia="Madrid" altitud="667">
    <dia fecha="2026-09-01"><tmax>31.2</tmax><tmin>18.4</tmin><lluvia>0</lluvia></dia>
    <dia fecha="2026-09-02"><tmax>29.8</tmax><tmin>17.9</tmin><lluvia>0</lluvia></dia>
    <dia fecha="2026-09-03"><tmax>24.1</tmax><tmin>15.2</tmin><lluvia>12.4</lluvia></dia>
    <dia fecha="2026-09-04"><tmax>26.5</tmax><tmin>14.8</tmin><lluvia>1.2</lluvia></dia>
  </estacion>
  <estacion id="E02" nombre="Navacerrada" provincia="Madrid" altitud="1894">
    <dia fecha="2026-09-01"><tmax>19.6</tmax><tmin>9.1</tmin><lluvia>0</lluvia></dia>
    <dia fecha="2026-09-02"><tmax>18.2</tmax><tmin>8.4</tmin><lluvia>3.5</lluvia></dia>
    <dia fecha="2026-09-03"><tmax>12.7</tmax><tmin>5.9</tmin><lluvia>28.1</lluvia></dia>
    <dia fecha="2026-09-04"><tmax>15.0</tmax><tmin>6.3</tmin><lluvia>6.0</lluvia></dia>
  </estacion>
  <estacion id="E03" nombre="Valencia Puerto" provincia="Valencia" altitud="5">
    <dia fecha="2026-09-01"><tmax>30.1</tmax><tmin>23.5</tmin><lluvia>0</lluvia></dia>
    <dia fecha="2026-09-02"><tmax>30.6</tmax><tmin>24.0</tmin><lluvia>0</lluvia></dia>
    <dia fecha="2026-09-03"><tmax>28.4</tmax><tmin>22.1</tmin><lluvia>0.4</lluvia></dia>
    <dia fecha="2026-09-04"><tmax>27.9</tmax><tmin>21.7</tmin><lluvia>15.8</lluvia></dia>
  </estacion>
</observaciones>
```

Apartados

- El nombre (el valor del atributo, como texto) de las estaciones situadas a más de 500 metros de altitud, en el orden del documento.
- La temperatura máxima más alta registrada en todo el periodo (un número).
- Las fechas (sin repetir y en el orden del documento) en las que alguna estación registró más de 10 mm de lluvia.
- Para cada estación, en el orden del documento, una cadena `nombre: media de máximas X °C, lluvia total Y mm`, con la media redondeada a un decimal.
- El día de mayor amplitud térmica (máxima menos mínima) de toda la red, como una única cadena `estación (fecha): amplitud X °C` con la amplitud redondeada a un decimal.
- Exporta a CSV (salida de texto) los días con lluvia: una primera línea `estacion;fecha;tmax;tmin;lluvia` y una línea por cada día con más de 0 mm, con el nombre de la estación, ordenadas por nombre de estación y después por fecha. Cada línea termina con un salto de línea.
- Escribe la DTD `observaciones.dtd`: `observaciones` con uno o más `estacion` y los atributos obligatorios `red` y `periodo`; `estacion` con uno o más `dia` y los atributos obligatorios `id` (de tipo ID, único en el documento), `nombre`, `provincia` y `altitud`; `dia` con `tmax`, `tmin` y `lluvia` en ese orden y el atributo obligatorio `fecha`; y `tmax`, `tmin` y `lluvia` con texto.

Soluciones

1. Catálogo de una editorial: XPath, XQuery, XSLT y XSD

Difícil · 90 minutos · apuntesdam.com/ejercicios/xml/catalogo-de-una-editorial-xml

Apartado 1 (xquery)

```
/catalogo/libro[@genero = 'novela']/titulo/text()
```

Apartado 2 (xquery)

```
/catalogo/libro[precio = max(/catalogo/libro/precio)]/titulo/text()
```

Apartado 3 (xquery)

```
count(/catalogo/libro[@stock = 0])
```

Apartado 4 (xquery)

```
/catalogo/libro[count(autor) > 1]/titulo/text()
```

Apartado 5 (xquery)

```
for $g in distinct-values(/catalogo/libro/@genero)
let $libros := /catalogo/libro[@genero = $g]
order by $g
return concat($g, ' (', count($libros), '): precio medio ', round(avg($libros/precio) * 100) div 100, '
€')
```

Apartado 6 (xquery)

```
for $l in /catalogo/libro[@stock > 10]
let $precio := round($l/precio * 0.8 * 100) div 100
order by $precio
return <oferta codigo="{ $l/@codigo }"><titulo>{string($l/titulo)}</titulo><precio>{$precio}</precio>
</oferta>
```

Apartado 7 (xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"/>
  <xsl:template match="/catalogo">
    <table>
      <tr><th>Titulo</th><th>Autores</th><th>Precio</th></tr>
      <xsl:for-each select="libro">
        <xsl:sort select="titulo"/>
        <tr>
          <xsl:if test="@stock = 0">
            <xsl:attribute name="class">agotado</xsl:attribute>
          </xsl:if>
          <td><xsl:value-of select="titulo"/></td>
          <td>
            <xsl:for-each select="autor">
              <xsl:value-of select="."/>
              <xsl:if test="position() != last()">,</xsl:if>
            </xsl:for-each>
          </td>
          <td><xsl:value-of select="format-number(precio, '0.00')"/> €</td>
        </tr>
      </xsl:for-each>
    </table>
  </xsl:template>
</xsl:stylesheet>
```

Apartado 8 (xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="catalogo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="libro" type="tipoLibro" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="editorial" type="xs:string" use="required"/>
      <xs:attribute name="actualizado" type="xs:date" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="tipoLibro">
    <xs:sequence>
      <xs:element name="titulo" type="xs:string"/>
      <xs:element name="autor" type="xs:string" maxOccurs="unbounded"/>
      <xs:element name="anio">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:minInclusive value="1450"/>
            <xs:maxInclusive value="2026"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="precio">
        <xs:simpleType>
          <xs:restriction base="xs:decimal">
            <xs:minInclusive value="0"/>
            <xs:fractionDigits value="2"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="paginas" type="xs:positiveInteger"/>
    </xs:sequence>
    <xs:attribute name="codigo" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="L[0-9]{3}"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="genero" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="novela"/>
          <xs:enumeration value="poesia"/>
          <xs:enumeration value="ensayo"/>
          <xs:enumeration value="infantil"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="stock" type="xs:nonNegativeInteger" use="required"/>
  </xs:complexType>
</xs:schema>
```

Las cuatro tecnologías comparten XPath: es el lenguaje para señalar partes de un documento, y aparece dentro de los `for` de XQuery, en los `select` y `test` de XSLT y en las restricciones de identidad de XSD. Dominar rutas y predicados es la mitad del trabajo con XML.

XQuery es a XML lo que SQL a las bases de datos relacionales: el FLWOR recuerda a `SELECT ... FROM ... WHERE ... ORDER BY`, con la ventaja de que el resultado puede ser XML nuevo. Eso lo hace útil para transformar datos entre sistemas.

XSLT resuelve el mismo problema desde otro ángulo: describe cómo debe ser la salida (aquí, HTML) a partir de plantillas. Por eso es habitual en la publicación de documentos y en la integración entre aplicaciones.

El XML Schema es el contrato del documento: con tipos y restricciones, cualquier librería puede validar el catálogo antes de cargarlo y rechazar un precio negativo o un género desconocido sin escribir código. Los documentos inválidos del ejercicio son justo los errores que el esquema tiene que atrapar.

2. Observaciones meteorológicas: XPath, XQuery, XSLT a CSV y DTD

Difícil · 80 minutos · apuntesdam.com/ejercicios/xml/observaciones-meteorologicas-xml

Apartado 1 (xquery)

```
/observaciones/estacion[@altitud > 500]/@nombre/string()
```

Apartado 2 (xquery)

```
max(//tmax)
```

Apartado 3 (xquery)

```
distinct-values(//dia[lluvia > 10]/@fecha)
```

Apartado 4 (xquery)

```
for $e in /observaciones/estacion
return concat($e/@nombre, ': media de máximas ', round(avg($e/dia/tmax) * 10) div 10, ' °C, lluvia total ', sum($e/dia/lluvia), ' mm')
```

Apartado 5 (xquery)

```
(for $d in //dia
let $amplitud := $d/tmax - $d/tmin
order by $amplitud descending
return concat($d/../@nombre, ' (', $d/@fecha, '): amplitud ', round($amplitud * 10) div 10, ' °C'))[1]
```

Apartado 6 (xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text"/>
  <xsl:template match="/observaciones">
    <xsl:text>estacion;fecha;tmax;tmin;lluvia&#10;</xsl:text>
    <xsl:for-each select="estacion/dia[lluvia > 0]">
      <xsl:sort select="../@nombre"/>
      <xsl:sort select="@fecha"/>
      <xsl:value-of select="concat(../@nombre, ';', @fecha, ';', tmax, ';', tmin, ';', lluvia)"/>
      <xsl:text>&#10;</xsl:text>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

Apartado 7 (xml)

```
<!ELEMENT observaciones (estacion+)>
<!ATTLIST observaciones
  red      CDATA #REQUIRED
  periodo  CDATA #REQUIRED>
<!ELEMENT estacion (dia+)>
<!ATTLIST estacion
  id      ID #REQUIRED
  nombre  CDATA #REQUIRED
  provincia CDATA #REQUIRED
  altitud CDATA #REQUIRED>
<!ELEMENT dia (tmax, tmin, lluvia)>
<!ATTLIST dia fecha CDATA #REQUIRED>
<!ELEMENT tmax (#PCDATA)>
<!ELEMENT tmin (#PCDATA)>
<!ELEMENT lluvia (#PCDATA)>
```

Las consultas XPath se apoyan en la conversión automática a número: comparar un atributo con 500 o un elemento con 10 funciona directamente, y las funciones de agregado (`max`, `avg`, `sum`) hacen el resto.

El eje padre (`..`) es lo que distingue a XPath de una búsqueda de texto: desde un día se llega a su estación sin guardar nada aparte. En XQuery, guardar la amplitud en un `let` permite ordenar por ella y usarla en el resultado.

La salida de texto de XSLT convierte XML en cualquier formato plano (CSV, ficheros de configuración, SQL). Ordenar por dos claves con dos `xsl:sort` funciona igual que un `ORDER BY` con dos columnas.

La DTD es más limitada que el XML Schema (no tiene tipos de datos: una altitud negativa sería válida), pero describe la estructura y garantiza identificadores únicos con `ID`. Por eso sigue apareciendo en documentos antiguos y en el `DOCTYPE` de HTML.