

## XML y su uso -Coloque Ejemplo

XML (eXtensible Markup Language) es un lenguaje de marcado para representar datos estructurados en texto plano. Permite definir tus propias etiquetas para describir la información; está pensado para almacenar y transportar datos de forma legible por humanos y máquinas.

### Características clave:

- Estructura jerárquica (árbol) con un único elemento raíz.
- Etiquetas definidas por el usuario (extensible).
- Puede validarse con DTD o XSD (esquemas).
- Muy soportado por bibliotecas y herramientas en todos los lenguajes.
- Legible pero verboso.

### Usos principales:

- Intercambio de datos entre sistemas (B2B, EDI).
- Servicios web basados en SOAP y WSDL.
- Archivos de configuración (p. ej. web.xml, app.config).
- Documentos y formatos estandarizados (Office Open XML, SVG, RSS/Atom).
- Transformaciones y presentaciones (XSLT).

### Ventajas / Desventajas (resumido)

**Ventajas:** flexible, autodescriptivo, soporta validación estricta, gran ecosistema.

**Desventajas:** verboso, más pesado y lento que JSON para APIs ligeras; mayor complejidad y vectores de seguridad (p. ej. XXE) si no se protege.

### Ejemplo práctico

Ejemplo de documento XML sencillo (una nota):

```
<?xml version="1.0" encoding="UTF-8"?>

<note>

  <to>María</to>

  <from>Juan</from>

  <heading type="reminder">Recordatorio</heading>

  <body>Compra pan y leche.</body>

</note>
```

### Explicación rápida del ejemplo

<?xml version="1.0"?> — declaración XML (versión/encoding).

<note> — elemento raíz que contiene todo.

<to>, <from>, <heading>, <body> — elementos hijos con contenido de texto.

type="reminder" — atributo del elemento heading.

### Ejemplo de validación (XSD muy básico)

Si quieres garantizar la estructura, defines un esquema (XSD) que valide que note tenga esos elementos:

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="note">

    <xs:complexType>

      <xs:sequence>

        <xs:element name="to" type="xs:string"/>

        <xs:element name="from" type="xs:string"/>

      </xs:sequence>

    </xs:complexType>

  </xs:element>

</xs:schema>
```

```
<xs:element name="heading">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="type" type="xs:string" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="body" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

### Ejemplo rápido de lectura en JavaScript (navegador)

```
const xmlString = `... (el XML anterior) ...`;
const parser = new DOMParser();
const doc = parser.parseFromString(xmlString, "application/xml");
const to = doc.querySelector("to").textContent; // "María"
```

### Recomendaciones de uso

- Usa XML cuando necesites validación estricta, interoperabilidad con sistemas empresariales o formatos estandarizados.
- Para APIs web modernas y móviles prefiera JSON por su ligereza, salvo que el ecosistema

demande XML (SOAP, formatos legados, documentos ofimáticos, etc.).

- Siempre valida y desactiva características peligrosas del parser (por ejemplo, resolución de entidades externas) para evitar vulnerabilidades XXE.