XML

Formato de datos

- Información desestructurada (unstructured)
  - la información no está organizada de una forma predefinida
    - Ejemplo
      - Archivo de texto, email, media
      - Corresponde con el 70-80% de la información en las organizaciones
- Información estructurada
  - modelo de datos
  - ejemplo
    - bases de datos
    - modelos semánticos
- Información semi-estructurada (semi-structured)
  - Marcadores (por ejemplo: etiquetas) para indicar la estructura
  - Estructura autodescriptiva (self-describing)
  - ejemplo
    - XML
    - JSON

Lenguajes de marcado

- Markup language
- Sistema para anotar documentos
- Distinguible del texto (contenido)
- No se muestra cuando se procesa
- Sirve para formatear el texto
- importa el orden
- Ejemplos
  - TeX (LaTeX)
  - HTML
  - SGML
    - Standard Generalized Markup Language
    - Estándar para definir lenguajes de marcado para documentos
    - Información estructural y semántica
    - No define la presentación del documento
    - Norma ISO 8879

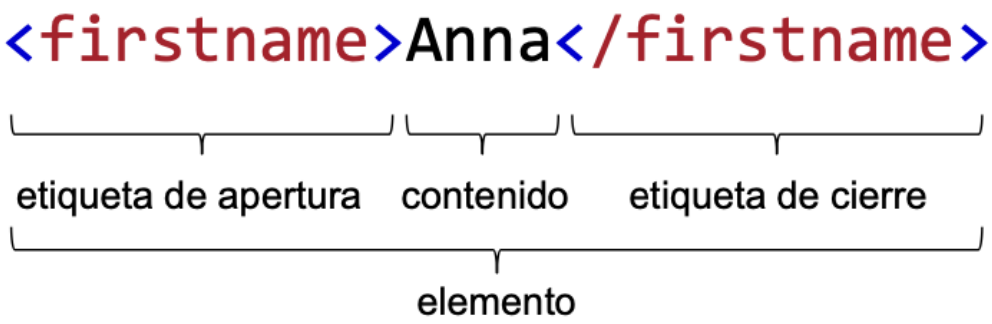
XML

- Extensible Markup Language
- Lenguaje de marcado con etiquetas
- Las etiquetas
  - describen el contenido
  - no describen la forma de mostrar el contenido
  - No están predefinidas

- Tiene estructura de árbol
- Estándar para la representación y envío de información
- Human and machine readable
- Es una extensión del SGML
- Usado en diferentes sectores
  - Negocios
  - Transacciones financieras
  - Datos médicos
  - Datos matemáticos
  - Mediciones científicas
  - Información de noticias
  - Servicios meteorológicos
- Es importante el orden
- Case sensitive
- Puede haber etiquetas repetidas
- Espacios y tabulados dan igual
- Puedes tener un elemento sin nada dentro
- No importa el orden para estar bien formado puede o no aparecer

Reglas XML

- Etiqueta
  - Marca sobre un elemento
  - Delimitada por < y >
  - se cierra con "/" destras de "<"
  - Exactamente el mismo nombre en la etiqueta de apertura y la de cierre
- Contenido
  - Zona entre la etiqueta de inicio y la de final
    - Ejemplo
      - <title>Contenido<title>
- Elemento
  - Etiqueta de inicio, contenido y etiqueta de cierre



- atributo
  - Información adicional asociada a una etiqueta
    - Formato: nombre = "valor"
    - el valor delimitado siempre por las comillas " " o apostrofes ' '
    - Cada elemento solo puede tener un atributo con el mismo nombre

        - siempre tiene un valor asociado
            - <book price="100">
- Comentarios
    - No se procesa
    - entre <!-- y -->
    - No puede contener "--"
- Se distinguen mayúsculas de minúsculas
- Todas las etiquetas abren y cierran
    - <book> </book>
    - <remark />
- Escapado caracteres especiales

&lt;	<	less-than	menor que
&gt;	>	greater-than	mayor que
&amp;	&	ampersand	et
&apos;	'	apostrophe	apóstrofe
&quot;	"	double-quote	comillas

- Nombre de las etiquetas y atributos
    - Case sensitive
    - Compuesto de caracteres Unicode
    - No puede contener: !"#$%&'()*+,/;<=>?@[\]^`{|}~
    - No puede contener espacios
    - No puede empezar por -, o número
- XML prolog
    - opcional
    - si existe, tiene que estar al principio, antes del elemento raíz
    - No es parte del documento XML
        - No es un error que no tenga etiqueta de cierre
    - <?xml version="1.0" encoding="UTF-8"?>
        - Define version XML
        - Define la codificación de caracteres
- Estructura de árbol
    - Document Object Model (DOM)
- Elemento raíz
    - root
    - tiene que ser único
- Todo los elementos excepto el raíz tienen padre (parent)
- Los elementos pueden tener hijos (Child)

- Los elementos que no tienen hijos son las hojas (leaves)
- Dos elementos que comparten padre son hermanos (sibling)

Elementos vs atributos

- Problemas de los atributos
  - No pueden contener múltiples valores
  - No se pueden expandir en un futuro
  - No pueden describir estructuras
  - se usan para expresar datos sobre datos (metadatos)
- Ventajas de los atributos
  - Legibilidad
  - Rendimiento
- En general, usar atributos únicamente para metadatos

Modelo relacional vs XML

	Modelo relacional	XML
Estructura	Tablas	Jerárquica en arbol
Esquema	Fijado con antelación	Flexible Auto-descriptivo
Consultas	Sencillas Intuitivas	Algo más complicadas
Ordenación	Ninguna	Implícita
Mejor para	Ordenador Almacenamiento	Humanos Streaming

XML Bien Formado

- Un documento XML está bien formado (well-formed) si se adhiere a unos requisitos estructurales básicos
  - Un único elemento raíz
  - Etiquetas emparejadas
  - Anidamiento apropiado
  - Atributos únicos dentro de cada elemento
  - Caracteres adecuados
- Analizadores (parsers)
  - Document Object Model (DOM)
  - Simple API for XML (SAX)
- Con XML no se permiten los errores

Mostrando XML

- Se utilizan lenguajes de reglas para transformar XML en HTML
  - CSS - Hojas de estilo en cascada
  - XSL - Lenguaje de hojas de estilo extensible

Estándares XML

- XML es el estándar de representación e intercambio de datos más extendido
- Cuenta con un gran número de estándares asociados
  - DTD
  - XSD
  - XPath
  - Xquery
  - XSL

XML Válido

- Documento XML bien formado (well - formed) => cumple requisitos estructurales básicos
- Un documento XML es válido (valid):
  - Está bien formado
  - Se adhiere a unos requisitos de contenido especificados por
    - DTD
      - Document Type Descriptor
    - XSD
      - XML Schema

DTD y XSD

- DTD
  - Document Type Descriptor
  - Descriptor de tipo de documento
  - Estándar para validar XML
  - Lenguaje que proporciona una gramática para especificar
    - Elementos
    - Atributos
    - Anidado
    - Ordenación
    - Número de apariciones

DTD - Elementos

- <!DOCTYPE **root-element** [...]>
  - **root-element:**
    - Nombre del elemento raíz
    - Tendrá que estar definido a continuación
- <!ELEMENT element-name category>
  - **element-name:** Nombre del elemento
  - **category:**

    - EMPTY: Vacío.
    - (#PCDATA): Parsed Character Data. Texto que se va a parsear
    - ANY: Cualquier combinación de datos parseables
    - (child1,child2,...):
        - Los hijos tienen que aparecer en el mismo orden
        - Se puede especificar el número de ocurrencias:
            - + : 1 or more
            - * : 0 or more
            - ? : 0 or 1
            - Si no se especifica, implica uno y sólo uno
        - Se puede usar | para permitir una elección
- importa el orden dentro del XML pero cuando escribimos el código del DTD no hace falta que este ordenado

Ejemplos:

```
<!ELEMENT name (#PCDATA)> → <name>Ana</name>
<!ELEMENT email (from,to+,body)> →
  <email>
    <from>…</from>
    <to>…</to>
    <to>…</to>
    <body>…</body>
  </email>
```

- <!ATTLIST element-name attribute-name attribute-type attribute-value [attribute-name attribute-type attribute-value]*>
    - **element-name:** Nombre de elemento al que pertenece
    - **attribute-name:** Nombre del atributo
    - **attribute-type:**
        - CDATA: Character Data. Texto que no se va a parsear
        - (*en1*|*en2*|..): Sólo se permiten esos valores específicos
        - ID
        - IDREF
    - **attribute-value:**
        - *value,*
        - #REQUIRED: El atributo tiene que estar
        - #IMPLIED: El atributo es opcional
    - [attribute-name attribute-type attribute-value]*
        - Se pueden añadir más atributos con la misma sintaxis

        - También se pueden definir en otra etiqueta ATTLIST

DTD Ejemplo con Punteros

- atributos ID y IDREF

UN XML ESTA BIEN FORMADO CUANDO ESTRUCTURA ES VALIDA

XSD

- XML Schema Definition
- También conocido simplemente como XML Schema
- Estándar para validar XML
- Basado en XML
- Lenguaje que proporciona una gramática para especificar
    - Elementos, orden, anidación y número de apariciones
    - Atributos
    - Tipos de datos y valores por defecto
- Elemento raíz
    - <xs:schema [archivo_xsd] ></xs:schema>
    - atributos
        - xmlns:**xs**=
            - especifica el namespace
            - todos los elementos y tipos de datos tienen que usar el prefijo xs
                - se puede cambiar
        - xs:schemaLocation = "Location"
            - localización del archivo xsd
        - [archivo_xsd]: En caso de definirse el xsd en otro archivo

XSD - Simple Element

- Un elemento que solo contiene tipos de datos simples
    - no puede contener otros elementos ni atributos
- Hay 19 tipos de datos primitivos
- 25 derivados
    - Se pueden componer para crear nuevos tipos de datos
        - Restringiendo
        - Listando una secuencia de valores permitidos
- Tipos de datos principales
    - xs:string
    - xs:decimal
    - xs:integer
- <xs: element name= "xxx" type= "yyy"/>
    - xs supone el prefijo definido en la etiqueta schema
    - xxx nombre del elemento
    - yyy tipo de dato
    - Otros atributos opcionales

- default = "valor"
  - valor por defecto si no se especifica uno
- fixed = "valor"
  - valor fijo que no se puede cambiar

```
<xs:element name="xxx" type="yyy"/>
```

- xs: suponiendo el prefijo xs definido en la etiqueta schema
- xxx: Nombre del elemento
- yyy: Tipo de datos
- Otros atributos opcionales:
  - `default="valor"`: Valor por defecto si no se especifica uno
  - `fixed="valor"`: Valor fijo que no se puede cambiar

Ejemplo

- Ejemplo de un fragmento XML:

```
<lastname>Fernández</lastname>
<dateborn>1970-03-27</dateborn>
<postal_code>28668</postal_code>
```

- Ejemplo de un fragmento XSD:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="dateborn" type="xs:date"/>
<xs:element name="postal_code" type="xs:integer"/>
```

Simple element restricciones

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
```

```
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Complex Element

- Contiene otros elementos y/o atributos
- Tipos
  - Vacíos
  - Elementos que solo contienen otros elementos
  - Elementos que solo contienen texto
  - Elementos con otros elementos y texto

```
<xs:attribute name="xxx" type="yyy"/>
```

- xxx: Nombre del atributo
- yyy: Tipo de dato
- Los atributos son opcionales por defecto
- Otros atributos opcionales:
  - `default="valor"`: Valor por defecto si no se especifica uno
  - `fixed="valor"`: Valor fijo que no se puede cambiar
  - `use="required"`: Para indicar que el atributo es obligatorio

También se le puede dar un nombre y definir aparte

```
<xs:element name="product" type="prodtype"/>

<xs:complexType name="prodtype">
  <xs:attribute name="prodid" type="xs:positiveInteger"/>
</xs:complexType>
```

<xs: sequence>

- Indica el orden en el que tienen que aparecer los elementos

<xs:extension base = "xs:integer">

- Para limitar el tipo de dato de texto

mixed = "true"

- Para permitir que el texto aparezca entre los elementos

```
<letter>
  Dear Mr. <name>John Smith</name>.
  Your order <orderid>1032</orderid>
  will be shipped on <shipdate>2021-07-13</shipdate>.
</letter>
```

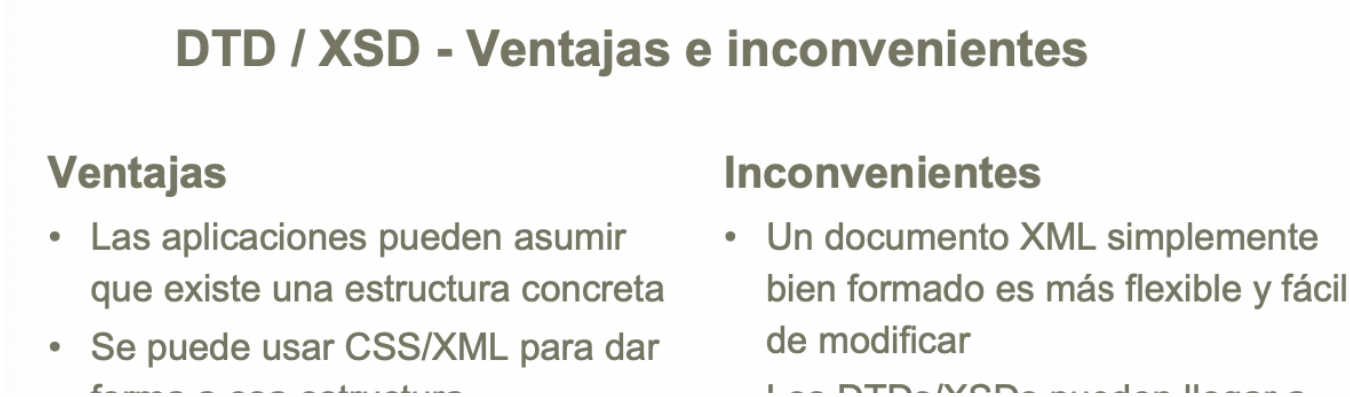
En vez de <xs:sequence> se pueden usar otros indicadores:

- <xs:all>
  - Los hijos pueden aparecer en cualquier orden
  - Cada hijo tiene que aparecer una vez
- <xs:choice>
  - solo puede aparecer uno de los hijos una vez

También se puede especificar la ocurrencia de cada elemento como atributo

- maxOccurs = "value"
  - Máximo número de ocurrencias
  - Por defecto 1
  - Se puede usar el valor ubounded para indicar que no hay límite
- minOccurs = "value"
  - Mínimo número de ocurrencias
  - Por defecto 1

DTD / XSD - Ventajas e inconvenientes



forma a esa estructura
- Es más fácil escribir documentación y hay una estructura fija
- Resto de ventajas del tipado fuerte

- Los DTDs/XSDs pueden llegar a ser muy complejos y difíciles de manipular
- Resto de desventajas del tipado débil

DTD / XSD - Comparativa

DTD	XSD
Document Type Definition	XML Schema Definition
Derived from SGML syntax	Written in XML
Doesn't support datatypes	Supports datatypes for elements and attributes
Doesn't support namespace	Supports namespace
Very limited control over number of elements	Can precisely define number of elements
Not extensible	Extensible
Not simple to learn	Simple to learn because you don't need to learn new language
Less control on XML structure	More control on XML structure
Doesn't allow duplicated definition of elements	Allows duplicated definition of elements

---

XPATH, XQUERY y XSLT

- XPath
  - Como navegamos en un documento XML?
    - Hay que pensar en el documento XML como en un árbol
      - Document Object Model (DOM)
    - Para navegar en dicho árbol utilizaremos XPath
  - XML Path Language
  - Sintaxis parecida a la de los directorios (paths)
  - Usado en otros estándares como XSLT
  - Recomendado por W3C

Tipos de nodos XPath

- Elemento
- Atributo
- Texto
- Namespace
- Instrucción a procesar
- Comentario
- Documento

XPath

- Construcción básica para rutas
  - / - Separador si empieza por /, referencia al nodo raíz
  - // - Yo y cualquier elemento descendiente
  - /Book - Etiquetas
  - /@ISBN - Atributos
    - Se obtiene su valor con /data(@ISBN)
  - | - OR lógico
  - * - Comodín

Xpath ejercicio
- /class
- //class
- /class//student
  - //student
  - //student/name
- /class/student
- /class/student/name
- /class/student/@id
- /class/student/data(@id)
- /class/student/data(@id|@otroAtributo)

XPath Predicados
- [ ]: Separador del predicado
  - Se pueden anidar para agrupar condiciones
- <,>, =, != : Comparadores
- and: AND lógico, enlaza condiciones
- or: OR lógico, enlaza condiciones
- [número]: Contador, se empieza a contar en 1
- [@attr]: Selecciona los atributos de nombre attr
- [@attr="value"]: atributos attr con valor value

//student[2]
//student[@id="035"]
//student/year
//student[year>1998]
//student[@*]

XPath Funciones
- Cuenta con multitud
- Se pueden usar en los predicados
  - Ejemplos
    - Contiene: contains (elemento, "texto")
    - Nombre: name()
    - Contador: count()

        - Contenido del nodo: text()
        - último elemento: last()

count(//student)
//student[year>"1998"]/name/text()
//student[name="Ana"]/year/text()
//student[last()]

Ejes de navegación en XPath

- Incluye 13 distintos
- Padre: parent::
- Hermano que le precede: preceding-sibling
- Hermano que le siga: following-sibling::
- Descendientes: descendant:
- Propia etiqueta: self:

XQuery

- Lenguaje para realizar consultas sobre XML
- Similar al concepto de SQL
- Version 3.1 en 2017
- Cada expresión opera sobre y devuelve una secuencia de elementos
    - Documento XML o stream XML
- Xpath es uno de los tipos de expresiones que soporta
    - Podemos incluir Xpath dentro de xquery

FLOWR

- For
    - Selecciona una secuencia de nodos
- let
    - enlaza la secuencia a una variable
- where
    - filtra los nodos
- order by
    - ordena los nodos
- return
    - lo que devolvemos
    - evaluado una vez por cada nodo

XQuery

- Con XQuery tenemos más opciones
- Podemos ordenar el resultado (archivo xquery_ejemplo1a.xq)

```
for $x in doc("xquery_ejemplo1.xml")/bookstore/book
where $x/price>30
order by $x/title
```

```
return $x/title
```

- Podemos incluir texto / XML / HTML

```
<ul> {
    for $x in doc("xquery_ejemplo1.xml")/bookstore/book/title
    order by $x
    return <li>{data($x)}</li>
} </ul>
```

XQuery vs XPath

XQuery	XPath
Functional programming and query language that is used to query a group of XML data	XML path language that is used to select nodes from an xml document using queries
Used to extract and manipulate data from either xml documents or relational databases and ms office documents that support an xml data source	Used to compute values like strings, numbers and boolean types from another xml documents
Represented in the form of a tree model with seven nodes, namely processing instructions, elements, document nodes, attributes, namespaces, text nodes, and comments	Represented as tree structure, navigate it by selecting different nodes
Supports Xpath and extended relational models	A component of query language
Helps to create syntax for new xml documents	Created to define a common syntax and behavior model for Xpointer and XSLT

XSLT

- XSL
  - Extensible Stylesheet Language
    - version inicial
- XSLT: XSL with Transformations
  - Version mejorada
- XSLT se escribe utilizando XML
- Estructura el documento en nodos
  - elementos, atributos, texto, comentarios
- Sirve para encontrar partes de un documento XML usando XPath y reemplazarlas por otras
- Busca conforme a una plantilla y reemplaza el resultado entero

XSLT

- Pueden aplicarse recursivamente
- Usa construcciones típicas de los lenguajes de programación
  - condicionales
    - if - else
  - Iteraciones
    - for-each
- Al usarlo se debe tener cuidado con
  - Comportamientos extraños con los espacios en blanco

    - Comportamientos extraños con los espacios en blanco
    - Prioridad implícita de las plantillas

XSLT - Declaración

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

- Hay que incluirlo al principio del documento (después del XML prolog, si existe)
- En la URL esta definido el estandar
- Con xmlns:xsl definimos xsl como el namespace que vamos a utilizar para tener acceso a las etiquetas definidas
- También podemos usar totalmente equivalente

XSLT - Elementos

```
<xsl:template match="pattern">...</ xsl:template>
```

- Para construir plantillas
- El contenido es lo que se escribe a la salida
- También se puede usar para descartar
- El atributo match la asocia a un nodo o nodos del XML
- El valor de match es una expresión XPath
- match="/" cubre todo el documento
- match="text()" cubre el texto, ni las etiquetas ni los atributos
- match="*|@*|text()" cubre todo el documento, pero cada ente por separado
- Cuidado con el orden de las plantillas

XSLT - Elementos

```
xsl:value-of select="expression" />
```

- Extrae el valor de un nodo
- El atributo select especifica lo que se extrae
- El valor de select es una expresión XPath
- <xsl:for-each select="expression">...</xsl:for-each>
- Extrae todos los valores de un nivel
- El atributo select especifica lo que se extrae
- El valor de select es una expresión XPath

XSLT - Elementos

```
<xsl:sort select="expression" />
```

- Ordena los valores
- En orden ascendente por defecto
    - Podemos añadir el atributo order="descending"
- Se usa dentro de un for-each
- El atributo select especifica lo que se extrae

- El atributo select especifica lo que se ordena
- El valor de select es una expresión XPath

XSLT - Elementos

```
<xsl:if test="expression"> ... </xsl:if>
```

- Impone una condición
- No permite else
  - Usar choose, when, otherwise
- Se usa dentro de un for-each
- El atributo test especifica lo que se comprueba
- Ejemplo: <xsl:if test="price > 10"> ... </xsl:if>

XSLT – Elementos

```
<xsl:attribute name="value">
  <xsl:value-of select="..." />
</xsl:attribute>
```

- Para añadir un atributo a un elemento

XSLT – Elementos

```
<xsl:variable name="nombreVariable" select="..."/>
```

- Para definir una variable
- Luego se utiliza con $nombreVariable

XSLT

```
<xsl:apply-templates select="expression"/>
```

- Permite encadenar templates
- En el select podemos especificar los elementos sobre los que se aplica
- Sirve para reutilizar código

XQuery	XSLT
Program driven	Function Driven
Declarative	Functional
Not XML based	XML based
Simple transformations	Complex transformations
Designed for retrieving and interpreting information	Mainly designed for transforming
Easier	Difficult
Shorter and faster	Hard to mantain

Node.js

- Node.js no está diseñado para trabajar con XML
- Pero puede ser necesario tener que usar XML
- Afortunadamente hay muchas librerías para trabajar con XML
    - En general parsean XML ↔ JSON
    - En general no validan
    - En general no permiten trabajar con XPath, XQuery ni con XSLT
- Librerías como
    - xmlbuilder
    - sax
    - fast-xml-parser

RSS

- RDF Site Summary
- Really simple syndication
- Estándar para feed de noticias
- info que cambia a menudo
- Feed web:
    - información que cambia a menudo
- Formato XML
- News/ Feed aggregators
    - Aplicaciones para procesar los RSS
    - Permiten seleccionar las feeds
    - Presentan al usuario la información de las feeds
    - Ejemplos
        - Feedly
        -