

# XML-DTD-SCHEMA

---

## Unidad de Trabajo 4:

Definición de esquemas y vocabularios en XML:



# XML-DTD-SCHEMA

---

Criterios de evaluación  
Contenidos

# Criterios de evaluación

---

- Se ha establecido la necesidad de describir la información transmitida en los documentos XML y sus reglas.
- Se han identificado las tecnologías relacionadas con la definición de documentos XML.
- Se ha analizado la estructura y sintaxis específica utilizada en la descripción.
- Se han creado descripciones de documentos XML.
- Se han utilizado descripciones en la elaboración y validación de documentos XML.
- Se han asociado las descripciones con los documentos.
- Se han utilizado herramientas específicas.
- Se han documentado las descripciones.



# Contenidos

---

- Estructura y sintaxis de documentos XML.
- DTD.
  - Declaración del tipo de documento.
  - Declaraciones de elementos.
  - Declaraciones de atributos.
  - Declaraciones de entidades.
  - Declaraciones de notaciones.
- XML Schema.
  - Elementos simples.
  - Restricciones.
  - Elementos complejos.

# XML

---

- ✓ Siglas en inglés de **eXtensible Markup Language** (lenguaje de marcas extensible).
- ✓ Propuesta para simplificar SGML.
- ✓ Desarrollado por el World Wide Web Consortium (W3C).
- ✓ Es un metalenguaje. Permite definir otros lenguajes.
- ✓ Pensado para describir datos, no para mostrar datos.
- ✓ Suficientemente flexible para ser utilizado en distintos ámbitos:
- ✓ XHTML
- ✓ RSS, Atom, OPML
- ✓ MathML            para notación matemática
- ✓ SVG                para describir gráficos vectoriales <https://developer.mozilla.org/en-US/docs/Web/SVG>
- ✓ KML                para representar datos geográficos  
<https://developers.google.com/kml/documentation/?hl=es>
- Lista de lenguajes de marcas [http://en.wikipedia.org/wiki/List\\_of\\_XML\\_markup\\_languages](http://en.wikipedia.org/wiki/List_of_XML_markup_languages)



# XML

---

- Por su parte, el éxito del XML ha sido enorme y cada vez es más utilizado como sistema de **intercambio** y **almacenamiento de información**.
- El W3C ha desarrollado alrededor del XML numerosas tecnologías para sacar provecho del XML.

# Características de XML

---

- Sus siglas provienen de eXtensible Markup Language.
- Puede almacenar y organizar cualquier tipo de información.
- Es un estándar internacional abierto (W3C).
- Utiliza Unicode por defecto . Cualquier idioma
- Permite revisar sintaxis y validar documentos.
- Es fácil de leer por personas y por aplicaciones o programas.
- Es posible exportarlo a numerosos formatos.



# XML

- Un documento XML es generalmente texto plano.
- Consta de etiquetas o marcas
- Marcas de inicio y de fin. Las de inicio delimitadas por `< y >` Las de fin delimitadas por `</ y >`
- Las etiquetas permiten definir elementos. `<etiqueta>Valor</etiqueta>`
- Cada elemento puede contener otros elementos.
- Los elementos pueden contener atributos. `<etiqueta atributo="valor">Valor</etiqueta>`
- En XML hay caracteres reservados
  - “        `&quot;`;
  - ‘        `&apos;`;
  - <       `&lt;`;
  - >       `&gt;`;
  - &       `&amp;`;



# Vocabulario

---

## **El procesador XML (XML processor) y aplicación (application)**

Cuando una aplicación necesita leer un documento XML, la aplicación recurre a un procesador XML.

**El procesador XML (o el analizador XML, en inglés XML parser)** es el que lee el documento, analiza el contenido y le pasa la información a un formato estructurado a la aplicación.

La recomendación XML especifica lo que debe hacer el procesador, pero no entra en lo que hace después la aplicación con esa información.

# XML

---

Instrucciones de procesamiento. Puede utilizarse para asociar una hoja de estilos.

**<?aplicación datos ?>**

Comentarios.

**<!-- comentario -->**

Secciones CDATA

**<![CDATA[“datos”]]>**

Los documentos XML forman una estructura de árbol.

**<raiz>**

**<padre>**

**<hijo> ..... </hijo>**

**<hermano> .... </hermano>**

**</padre>**

**</raiz>**



# Reglas XML

---

- Todos los elementos XML deben tener una etiqueta de apertura y otra de cierre.
- Si el elemento no tiene contenido podremos sustituir las etiquetas por `<etiqueta />`.
- Las etiquetas XML distinguen entre mayúsculas y minúsculas.
- En XML todos los elementos deben estar anidados correctamente. No pueden estar entrelazados.
- Los documentos XML deben tener un único elemento raíz.
- En XML los elementos pueden contener atributos en la etiqueta inicial.
- En XML los valores de los atributos deben ir entre comillas (simples o dobles).
- En XML un elemento no puede tener dos atributos con el mismo nombre.
- En XML se deben sustituir los caracteres reservados por sus respectivas entidades.
- Los comentarios y las instrucciones de procesamiento no pueden estar dentro de etiquetas.

# Elementos XML

---

- Un elemento está formado por la etiqueta de inicio, la etiqueta de fin y todo lo contenido entre ellas.
- Podrá contener: Texto, otros elementos y atributos
- Reglas para los nombres de elementos y atributos:
  - Pueden contener letras, números y otros caracteres (guion bajo, guion, punto).
  - No pueden contener un espacio en blanco.
  - Deben comenzar por letras o el carácter de subrayado.
  - No pueden comenzar por un número, un guion o un punto.
  - No pueden comenzar por xml (ya sea en minúsculas o en mayúsculas).
  - No hay limitación en cuanto a la longitud.
- Evitar en la medida de lo posible el uso de atributos.
- Utilizar los atributos para los metadatos.



# XML bien formado

---

- Los documentos bien formados son aquellos que son sintácticamente correctos.
- Contiene un único elemento raíz.
- Todas las etiquetas están correctamente anidadas.
- **XML válido**
  1. Debe estar bien formado.
  2. La estructura debe encajar con la definición del tipo de documento (DTD) o esquema (XML Schema, Relax NG).
- **Se comprobará:**
  1. Que elementos y atributos se permiten.
  2. Estructura de los elementos y los atributos.
  3. Orden de los elementos.
  4. Valores de los datos de los elementos y los atributos.
  5. Unicidad de valores dentro de un documento.

# Declaración XML

---

- Los documentos XML deben empezar con una declaración XML.  
`<?xml version="1.0" encoding="UTF-8" standalone="yes"?>`
- Debe estar al principio del documento.
  - El atributo **version** debe tener el valor 1.0.  
(<http://www.w3.org/TR/REC-xml/>)
  - El atributo **encoding** nos permite indicar la codificación de caracteres utilizada en el documento. Es opcional. Si se omite utiliza el conjunto de caracteres Unicode.
  - El atributo **standalone** indica si el documento es independiente o se basa en la información de fuentes externas. Es opcional. Si se omite se supone que el valor es **no**. Si el valor es **yes** puede requerirse un DTD externo.



# Ejemplo de documento XML

---

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

PROLOGO

```
<alumno>  
  <nombre>Francisco</nombre>  
  <apellido>Rodríguez</apellido>  
  <direccion>  
    <calle>Federico Silva</calle>  
    <cp>49600</cp>  
  </direccion>  
</alumno>
```

CUERPO

# Ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE documento SYSTEM "01.dtd">
<!-- Esto es un comentario -->
<documento>
  <p>Mi primer <negrita valor="1">documento XML</negrita></p>
  <p>Comienza con la etiqueta <lt;documento></p>
  <p>A continuación añadimos una etiqueta de documento vacio</p>
  <imagen archivo="imagen.png"/>
  <p>Añadimos una etiqueta CDATA</p>
  <ejemplo>
    <![CDATA[ Esto es una sección de datos en la que podemos
    escribir sin que el parser lo analice]]>
  </ejemplo>
</documento>
```



# Ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN" "dockbookx.dtd">
<!--Nuestro primer capítulo del libro -->
<chapter lang="es" id="capitulo_1">
  <title>Introducción a XML</title>
  <sect1>
    <title>Definición de XML</title>
    <para>
      <application>XML</application> es un lenguaje de descripción de documentos
    </para>
  </sect1>
</chapter>
```

# Etiquetas

---

Existen seis tipos de etiquetas: elementos, referencias a entidades, comentarios, instrucciones de procesamiento, secciones de datos y declaración de tipo de documento.

- **Elementos:** están delimitados por ángulos (<,>) e identifican el contenido que delimitan. Pueden tener atributos. Siguen la estructura:

`<elemento atributo="valor">`

`<chapter lang="es" id="capitulo_1">`



# Etiquetas

- **Referencias a entidades:** empiezan por “&” y acaban por “.”
- **Comentarios:** no forman parte del texto. Siguen la estructura: `<!-- comentario -->`

`<!--Nuestro primer capítulo del libro -->`

- **Instrucciones de procesamiento:** se utilizan para proporcionar información en un documento XML. El parser pasa esa información a la aplicación que realiza la llamada.
- `<?xml version="1.0" encoding="utf-8"?>`

# Etiquetas

---

- **Declaración de tipo de documento:** En este ejemplo, la instrucción de procesamiento indica que el documento XML se debe mostrar utilizando una hoja de estilos XSL:

```
<?xml-stylesheet type="text/xsl"
href="/observatorio/web/MyStyleSheet.xsl"?>
```

- Sección de datos: indica al parser que ignore el contenido de la sección. Siguen la estructura: `<![CDATA[texto]]>`

```
<![CDATA[ Esto es una sección de datos en la que podemos escribir sin que el parse lo analice, ]]>
```



# Conceptos y vocabulario

---

- **Referencia a entidades**

- Una entidad consiste en un nombre y su valor (son similares a las constantes en los lenguajes de programación). Las entidades se definen mediante la ENTITY,
- Por ejemplo, `<!ENTITY elcid "Rodrigo Díaz de Vivar">`
- Una referencia a una entidad empieza con el carácter "&", sigue con el nombre de la entidad y termina con ";". Al abrir el documento XML el procesador sustituye la referencia a la entidad por su valor.
- Por ejemplo, la etiqueta: `<autor>&elcid;</autor>`
- El procesador XML la convertiría en:
- `<autor>Rodrigo Díaz de Vivar </autor>`

# Ejemplo de uso espacio de nombres

```
<?xml version="1.0"?>
<html:html xmlns:html='http://www.w3.org/1999/xhtml'>
  <html:head>
    <html:title>Librería virtual</html:title>
  </html:head>
  <html:body>
    <html:p>Mover a
      <html:a href="http://vlib.example.org">vlib.example.org</html:a>
    </html:p>
  </html:body>
</html:html>
```

La declaración del espacio de nombres se aplica al elemento en que está especificada y a todos los elementos pertenecientes al contenido de ese elemento.



# Espacio de nombres

- En XML los nombres de los elementos están definidos por el desarrollador.
- Muchas veces surgen conflictos con nombres de elementos al intentar mezclar documentos XML de distintas fuentes.
- Para evitar el problema se le añade un prefijo al elemento que identifica el espacio de nombres.
- Para poder utilizar el prefijo debe ser definido el espacio de nombres.
- El espacio de nombres es definido por el atributo **xmlns** en la etiqueta inicial de un elemento.

**xmlns:prefijo="URI"**

- El espacio de nombres se puede declarar en los elementos donde se utiliza o en el elemento raíz.
- Recomendación W3C sobre los espacios de nombres en XML

**<http://www.w3.org/TR/REC-xml-names/>**

# Espacio de nombres en los esquemas

```
<?xml version="1.0"? encoding="UTF-8"?>
<xs:schema version="1.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.festivalgijon.com"
  targetNamespace="http://www.festivalgijon.com"
  elementFormDefault="qualified">
```

- **xmlns:xs="http://www.w3.org/2001/XMLSchema"** -> indica que los argumentos y tipos de datos usados (schema, element, complexType, sequence ,string,...) provienen del espacio de nombres <http://www.w3.org/2001/XMLSchema>.
- **targetNamespace="http://www.festivalgijon.com"** -> indica que los elementos definidos en este esquema (pelicula, url, titulo,..) pertenece al espacio de nombres indicado, que llamaremos de destino (target) y que obviamente se pasa al documento instancia para su uso correspondiente.
- **xmlns="http://www.festivalgijon.com"** -> indica el espacio de nombres por defecto del esquema, de forma que cuando aparece una referencia como ref="pelicula" sin ningún prefijo, se sabe que se refiere a la declaración de pelicula de este espacio de nombres.



# Ejemplo de uso de espacio de nombres

- Se pueden declarar varios prefijos de espacios de nombres como atributos de un mismo elemento.
- Se considera que se aplica un espacio de nombres por defecto al elemento en que está declarado (si ese elemento no tiene prefijo de espacio de nombres), y a todos los elementos sin prefijo pertenecientes al contenido de ese elemento.

```
<?xml version="1.0"?>  
  
<bk:book  
  xmlns:bk='urn:loc.gov:books'  
  xmlns:isbn='urn:ISBN:0-395-36341-6'>  
  
  <bk:title>Cheaper by the Dozen</bk:title>  
  
  <isbn:number>1568491379</isbn:numero>  
  
</bk:book>
```

# Ejemplo de uso de espacio de nombres

---

```
<?xml version="1.0"?>
<book xmlns='urn:loc.gov:books'
      xmlns:isbn='urn:ISBN:0-395-36341-6'>

  <title>Cheaper by the Dozen</title>
  <isbn:number>1568491379</isbn:number>
  <notes>
    <p xmlns='http://www.w3.org/1999/xhtml'>
      This is a <i>funny</i> book!
    </p>
  </notes>
</book>
```



# DTD

---

- **Document Type Definition** (definición de tipo de documento).
- Especifica que elementos pueden aparecer en un documento y dónde, así como el contenido y los atributos.
- Un documento válido incluye una declaración de tipo de documento que identifica la DTD que satisface el documento.
- Hay que distinguir entre declaración de tipo de documento y definición de tipo de documento.
- La DTD presenta una lista de todos los elementos, atributos y entidades que utiliza el documento.
- Cuando un documento tiene una declaración de tipo de documento y el documento se ajusta a la DTD se dice que el documento es válido.

# DTD. Declaración de tipo de documento

- Un documento válido incluye una referencia a la DTD con la que se debe comparar.
- Se incluye en el prólogo del documento XML, después de la declaración XML.
- Puede tratarse de una referencia interna o de una referencia externa.
- Sintaxis para declaración interna:

**<!DOCTYPE elemento\_raiz [ declaración de elementos ]>**

- Sintaxis para declaración externa:

**<!DOCTYPE elemento\_raiz SYSTEM "url\_dtd">**

**<!DOCTYPE elemento\_raiz PUBLIC "identificador\_publico" "url\_dtd">**

- Existe la posibilidad de que haya declaración interna y externa a la vez.

**<!DOCTYPE elemento\_raiz SYSTEM "url\_dtd" [ declaración de elementos ]>**



# DTD. Declaración de elementos

---

- Todos los elementos usados en un documento válido, deben declararse en la DTD de un documento con una declaración de elemento.

- Sintaxis:

**<!ELEMENT nombre\_del\_elemento  
especificación\_de\_contenido>**

- El nombre del elemento puede ser cualquier nombre XML.
- Especificación de contenido:
  - Datos de tipo carácter **(#PCDATA)**
  - Elementos hijos **(nombre\_elemento\_hijo)**
  - Elemento vacío **EMPTY**
  - Sin restricciones **ANY**

# DTD. Declaración de elementos

---

Especificación de contenido:

- Secuencias.

- Un elemento suele tener más de un hijo.

**(nombre\_elemento\_hijo1, nombre\_elemento\_hijo2)**

- Opciones

- Las instancias de un elemento pueden contener hijos distintos.

**(nombre\_elemento\_hijo1 | nombre\_elemento\_hijo2)**



# DTD

---

- Declaraciones de elementos:
- Especificación de contenido:
- Número de hijos.
- No todas las instancias de un elemento tienen que tener los mismos hijos.
- **Sufijos a añadir al nombre del elemento hijo:**
- ? el elemento hijo puede aparecer cero o una veces  
(nombre\_elemento\_hijo?)
- \* el elemento hijo puede aparecer cero o más veces  
(nombre\_elemento\_hijo\*)
- + el elemento hijo puede aparecer una o más veces  
(nombre\_elemento\_hijo+)

# DTD. Declaración de elementos

---

Especificación de contenido:

- Paréntesis.
- Las secuencias, las opciones y los sufijos suelen aparecer combinados.
- Una secuencia o una opción pueden aparecer entre paréntesis y tener un sufijo.
- Los paréntesis pueden estar anidados.
- Contenido mixto (`#PCDATA | nombre_elemento_hijo`)\*



# DTD . Ejemplo “Declaración interna”

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

<!DOCTYPE alumno [
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT apellido (#PCDATA)>
    <!ELEMENT calle (#PCDATA)>
    <!ELEMENT cp (#PCDATA)>
    <!ELEMENT direccion (calle, cp)>
    <!ELEMENT alumno (nombre, apellido, direccion)>
]>

<alumno>

    <nombre>Francisco</nombre>
    <apellido>Rodríguez</apellido>
    <direccion>
        <calle>Federico Silva</calle>
        <cp>49600</cp>
    </direccion>
</alumno>
```

# TD. Ejemplo “Declaración externa”

Archivo **alumno.dtd**

```
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellido (#PCDATA)>
<!ELEMENT calle (#PCDATA)>
<!ELEMENT cp (#PCDATA)>
<!ELEMENT direccion (calle, cp)>
<!ELEMENT alumno (nombre, apellido, direccion)>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE alumno SYSTEM "alumno.dtd">
<alumno>
  <nombre>Francisco</nombre>
  <apellido>Rodríguez</apellido>
  <direccion>
    <calle>Federico Silva</calle>
    <cp>49600</cp>
  </direccion>
</alumno>
```



# DTD. Declaraciones de atributos

---

- Además de declarar sus elementos, un documento válido tiene que declarar todos los atributos de dichos elementos.
- **Sintaxis:**

```
<!ATTLIST elemento nombre_atributo tipo_atributo  
                valor_predeterminado>
```

- Una sola declaración ATTLIST puede declarar múltiples atributos para el mismo elemento.
- Si el mismo atributo se repite en varios elementos, debe declararse en cada elemento donde aparece.

Valores predeterminados:

- **#IMPLIED** El atributo es opcional. No se proporciona valor predeterminado.
- **#REQUIRED** El atributo es obligatorio. No se proporciona valor predeterminado.
- **#FIXED** El valor del atributo no puede cambiar. Aparece entre comillas.
- **Valor literal** El valor predeterminado aparece entre comillas.

# DTD. Declaraciones de atributos

---

Tipos de atributo:

<b>CDATA</b>	Puede contener cualquier cadena de texto
<b>NMTOKEN</b>	Debe contener los mismos caracteres que un nombre XML
<b>NMTOKENS</b>	Contiene uno o más NMTOKEN separados por un espacio en blanco
<b>ID</b>	Debe contener un nombre XML único dentro del documento Cada elemento solo puede tener un atributo ID
<b>IDREF</b>	Debe ser un nombre XML Se refiere al atributo de tipo ID de algún elemento en el documento Se utilizan para establecen relaciones entre los elementos
<b>IDREFS</b>	Contiene una lista de nombres XML separados por un espacio en blanco Cada uno de los nombres XML se refiere al ID de un elemento



# DTD. Declaraciones de atributos

---

## Tipos de atributo:

- **ENTITY** Contiene el nombre de una entidad sin analizar, declarada en alguna parte en la DTD.
- **ENTITIES** Contiene los nombres de una o más entidades sin analizar, declaradas en cualquier parte en la DTD, separados por un espacio en blanco.
- **NOTATION** Contiene el nombre de una notación declarada en la DTD del documento
- **Enumeración** Lista de todos los valores posibles para el atributo, separados por barras verticales. Cada valor debe tener los mismos caracteres que un nombre XML.

```
<!ATTLIST elemento atributo (valor1 | valor2 | valor3)  
                        valor_predeterminado>
```

# DTD. Declaración de entidades (ENTITY)

---

- XML predefine cinco entidades:

**&lt;** ( < ) **&gt;** ( > ) **&amp;** ( & ) **&quot;** ( “ ” )  
**&apos;** ( ' )

- Las referencias de entidad se definen con la declaración ENTITY.
- Una entidad ofrece una entrada abreviada para el documento XML o la DTD.
- Las entidades se clasifican en:
  - Internas o externas
  - Analizadas o no analizadas
  - Generales o Parámetro



# DTD. Declaraciones de entidades (ENTITY)

---

## Entidades generales internas:

- Son abreviaturas definidas en la DTD del documento XML.
- Consta de nombre de la entidad y texto de reemplazo de la entidad.
- El nombre de la entidad deber ser un nombre XML.
- Son siempre entidades analizadas.
- Para referenciar la entidad en el documento XML: **&nombre\_entidad;**
- Una vez reemplazada la referencia a la entidad por su contenido, pasa a ser parte del documento XML y es analizada por el procesador XML.

- Ejemplo:

```
<!ENTITY nombre "texto de sustitución">
```

```
<etiqueta>Contenido: &nombre;</etiqueta>
```

# DTD. Declaraciones de entidades (ENTITY)

---

## Entidades generales externas analizadas:

- Obtienen su contenido de otro sitio.
- Consta de nombre de la entidad, la palabra SYSTEM y el URI (Universal Resource Identifier) .
- El nombre de la entidad deber ser un nombre XML.
- La entidad externa no contendrá prólogo, es decir, ni declaración XML ni declaración de tipo de documento.
- Puede que el analizador no reemplace la referencia de la entidad con el documento en el URI.
- Ejemplo:

```
<!ENTITY nombre SYSTEM "archivo.xml">  
&nombre;
```



# DTD. Declaraciones de entidades(ENTITY)

---

## Entidades no analizadas:

- Si el contenido de la entidad es un archivo de cualquier tipo no XML, el procesador XML no debe analizarlo.
- Siempre son entidades generales y externas.
- Consta de nombre de la entidad, la palabra SYSTEM y el URI (Universal Resource Identifier) .
- El nombre de la entidad deber ser un nombre XML.
- Una entidad sin analizar no puede referenciarse . Las referencias de entidad se usan sólo con entidades analizadas.
- XML no garantiza ningún comportamiento determinado de ninguna aplicación que se encuentre entidades sin analizar.

## Ejemplo:

```
<!ENTITY nombre SYSTEM "logo.jpg" NDATA  jpg>
```

# DTD. Declaraciones de entidades (ENTITY)

---

## Entidades de parámetro internas:

- Se utilizan en la DTD y no en el documento XML.
- Se pueden utilizar para agrupar ciertos elementos de la DTD que se repiten mucho.
- Se declara de forma parecida a la referencia de entidad general.
- Aparece el carácter % entre ENTITY y el nombre de la entidad.
- Las referencias de entidades de parámetro sustituyen el carácter & por el carácter %.
- Ejemplo:

```
<!ENTITY % nombre "<!ELEMENT .....>">  
%nombre;
```



# DTD. Declaración de entidades (ENTITY)

---

## Entidades de parámetro externas:

- Se utilizan en la DTD y no en el documento XML.
- No pueden ser utilizadas en DTD internas.
- Se pueden utilizar para agrupar ciertos elementos de la DTD que se repiten mucho.
- Se declara de forma parecida a la referencia de entidad general.
- Aparece el carácter % entre ENTITY y el nombre de la entidad.
- Las referencias de entidades de parámetro sustituyen el carácter & por el carácter %.
- Ejemplo:

```
<!ENTITY % nombre SYSTEM "archivo.dtd">  
%nombre;
```

# DTD. Declaraciones de notaciones (NOTATION)

---

- Permiten definir el formato de archivos externos no XML.
- Ejemplo:

```
<!NOTATION  jpg SYSTEM "image/jpeg">
```

```
<!ENTITY nombre SYSTEM "logo.jpg" NDATA jpg>
```

```
<!ELEMENT imagen EMPTY>
```

```
<!ATTLIST imagen img ENTITY #REQUIRED>
```

```
<imagen img="nombre" />
```



# XML Schema

- Documento XML que contiene una descripción formal de lo que comprende un documento XML válido. Desarrollado por el W3C (World Wide Web Consortium).
  - <http://www.w3.org/TR/xmlschema-0/>
  - <http://www.w3.org/TR/xmlschema-1/>
  - <http://www.w3.org/TR/xmlschema-2/>
- Un documento XML se describe mediante un esquema denominado documento de instancia.
- Si un documento satisface todas las restricciones especificadas por el esquema, se considera que es un documento con un esquema válido.
- El documento del esquema se asocia a un documento de instancia a través de uno de los siguientes métodos:
  - Un atributo **xsi:schemaLocation** en un elemento contiene una lista de espacios de nombre usados dentro de dicho elemento y los URL de los esquemas con los que se validan los elementos y atributos en dichos espacios de nombre.
  - Un atributo **xsi:noNamespaceSchemaLocation** contiene un URL para el esquema usado para validar elementos que no se encuentran en ningún espacio de nombre.



# XML Schema

---

- Los elementos utilizados en la creación de un esquema pertenecen al siguiente espacio de nombres:

**<http://www.w3.org/2001/XMLSchema>**

- El esquema del XML Schema: <http://www.w3.org/2001/XMLSchema.xsd>
- Todo documento de esquema contiene un solo elemento raíz **xs:schema**. Contendrá declaraciones para todos los elementos y atributos que pueden aparecer en un documento de instancia válido.
- A continuación puede aparecer el elemento opcional **xs:annotation**. Permite incluir información sobre autoría, usos, derechos de autor, etc...
- El elemento **xs:annotation** puede contener cualquier combinación de los elementos **xs:documentation** y **xs:appinfo**.
- El elemento **xs:documentation** se utiliza para contener información adicional legible para las personas. Admite un atributo **xml:lang** para identificar el idioma.
- El elemento **xs:appinfo** se utiliza para contener información adicional legible para las máquinas



# XML Schema

- El elemento **xs:element** se utiliza para definir los elementos que puede contener el documento XML.
- Un documento XML puede contener elementos de dos tipos: simples y complejos.
- Los elementos simples son aquellos que solo contienen texto. No contienen otros elementos ni atributos.
- Para definir un tipo simple el elemento **xs:element** podrá contener entre otros los siguientes atributos:
- **name** nombre del elemento
- **type** tipo del elemento (<http://www.w3.org/TR/xmlschema-2/>)
- **xs:string**                      **xs:integer**                      **xs:date**                      **xs:boolean**
- **minOccurs** mínimo número de veces que puede aparecer el elemento (defecto 1)
- **maxOccurs** máximo número de veces que puede aparecer el elemento (defecto 1)
- **default** valor por defecto si no se especifica otro valor para el elemento.
- **fixed** valor que se asigna al elemento y que no puede ser modificado.

# XML Schema- Elementos simples

---

- A los tipos simples se le pueden aplicar facetas (restricciones).
- Las facetas se aplican a tipos simples usando el elemento **xs:restriction**. Cada faceta se expresa como un elemento distinto dentro del bloque de restricciones.

```
<xs:element name="edad">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0">
        <xs:maxInclusive value="16">
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
```



# XML Schema - Restricciones

---

- Facetas numéricas:

**minInclusive**                      **minExclusive**                      **maxInclusive**  
**maxExclusive**  
**totalDigits**                      **fractionDigits**

- Enumeraciones:                      **enumeration**

- Restricciones de longitud:

**length**                      **minLength**                      **maxLength**

- Tratamiento de los espacios en blanco:

**whiteSpace** (valores posibles: preserve, replace, collapse)

- Forzar el formato:

**pattern** (el valor debe ser una expresión regular)

# XML Schema- Elementos complejos

---

- Los elementos complejos son aquellos que pueden contener atributos, otros elementos y texto. También los elementos vacíos.
- Para definir un tipo complejo deberemos utilizar el elemento **xs:complexType**.
- El elemento **xs:complexType** puede tener el atributo **name**.
- El elemento **xs:complexType** podrá contener los **elementos xs:sequence, xs:choice, xs:all**.
  - El elemento **xs:sequence** permite indicar el orden en el que deben aparecer los elementos.
  - El elemento **xs:choice** permite indicar una lista de elementos a elegir.
  - El elemento **xs:all** permite indicar elementos opcionales o que aparecen una sola vez. Pueden aparecer en cualquier orden.



# XML- Elementos complejos

---

Elementos que contienen otros elementos:

```
<xs:element name="direccion">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="calle" type="xs:string" />
      <xs:element name="cp" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="direccion" type="dire"/>
<xs:complexType name="dire">
  <xs:sequence>
    <xs:element name="calle" type="xs:string"/>
    <xs:element name="cp" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

# XML Schema- Elementos complejos

---

- Elementos que contienen otros elementos:

```
<xs:element name="calle" type="xs:string"/>
<xs:simpleType name="codpos">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:element name="direccion" type="dire"/>
<xs:complexType name="dire">
  <xs:sequence>
    <xs:element ref="calle"/>
    <xs:element name="cp" type="codpos"/>
  </xs:sequence>
</xs:complexType>
```

- Elementos vacíos:
- Debemos declarar los atributos mediante xs:attribute.



# XML Schema- Elementos complejos

- Elementos que contienen texto:

```
<xs:element name="nombre">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string"/>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

- Elementos que contienen otros elementos y texto:
- El atributo **mixed** de **xs:complexType** debe tener el valor true.

# XML Schema- Atributos

---

- Para la declaración de atributos se utiliza el elemento **xs:attribute**.
- El elemento **xs:attribute** podrá contener los siguientes atributos:
  - **name** nombre del atributo
  - **type** tipo del atributo (los atributos solo pueden contener tipos simples)
  - **use** Valores: **required**, **optional**, **prohibited**
  - **default** Si el atributo no aparece en el documento se le asigna el valor contenido
  - **fixed** El valor del atributo en el documento debe ser el mismo que el valor



# XML Schema.

---

```
<?xml version="1.0" encoding="UTF-8"?>
<alumno
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation= "esquema.xsd">

  <nombre>Francisco</nombre>
  <apellido>Rodríguez</apellido>
  <direccion>
    <calle>Federico Silva</calle>
    <cp>49600</cp>
  </direccion>
</alumno>
```

# XML Schema

---

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" >
  <xs:element name="alumno">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nombre" type="xs:string" />
        <xs:element name="apellido" type="xs:string" />
        <xs:element name="direccion">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="calle" type="xs:string" />
              <xs:element name="cp" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```