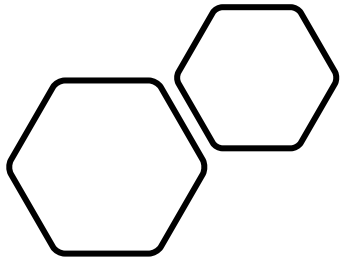
Curso 2023-2024



UD4: XML, validación y acceso

Lenguajes de marcas y sistemas de gestión de información

XML

- XML es, junto con HTML, uno de los lenguajes de marcas más estandarizados y característicos.
- XML es un metalenguaje, es decir, define las reglas de construcción de documentos, pero no especifica las etiquetas. Por tanto, en XML, en vez de tener elementos y atributos predefinidos, tenemos mecanismos para definirlos.
- XML está estandarizado por W3C, y esto ha favorecido la aparición de diversas tecnologías y herramientas para trabajar con este lenguaje: mecanismos de validación, editores, programas de visualización o librerías de programación para integrarlo en nuestro software.

XML

- El propósito de XML es más difuso que el de HTML. De forma general podemos decir que su principal propósito es el de intercambio de información, pero también se puede usar para definir interfaces de usuario, almacenar datos, guardar configuraciones, etc.
- XML fue lanzado en 1998 y actualmente se encuentra en la versión 1.1, pero W3C recomienda crear los documentos en la versión 1.0.

Estructura y sintaxis XML

- Todo documento XML se estructura en dos partes: prólogo y cuerpo.
- El prólogo contiene información relativa al conjunto del documento.
- El cuerpo recoge los elementos con la información propiamente dicha. También permite incluir comentarios que no serán procesados.
- La declaración del documento XML se realiza en el prólogo, siendo éste opcional. Si existe, debe ser la primera línea del documento.

Declaración del documento

- En él se indica que el documento es XML, la versión y la codificación mediante los siguientes atributos:
- version: la versión de XML, generalmente 1.0.
- encoding: la codificación usada en el documento.
- standalone: Indica si el documento XML es independiente o si existe un **DTD** (*Document Type Definition*) externo para realizar la **validación**. Admite los valores **yes** y **no**, y es un atributo opcional.

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

Sintaxis. Elementos y atributos

- El cuerpo de un documento XML está formado por elementos, similares a los que usamos en HTML: tienen una etiqueta de apertura y otra de cierre.
- La etiqueta de cierre es **obligatoria**. Es posible tener etiquetas “terminales”, en las que la apertura y el cierre se realizan en la misma:

```
<libro atributo="valor"> </libro>
<libro atributo="valor"/>
<libro atributo="valor">
```

Correcto

Correcto

Incorrecto

Sintaxis. Elementos y atributos

- Las reglas para la creación de **nombres de elementos** son:
  - Se diferencia entre mayúsculas y minúsculas.
  - Deben comenzar con una letra o un guion bajo.
  - El nombre de apertura tiene que ser idéntico al de cierre.
  - El nombre puede estar formado por caracteres alfanuméricos, guiones, guiones bajos y puntos.
  - Los nombres no pueden contener espacios (interpreta como nombre del elemento la primera palabra y el resto como parámetros).

Sintaxis. Elementos y atributos

- Las reglas para la creación de **atributos** son:
  - Deben tener asignado un valor.
  - Los valores del atributo van siempre entre comillas dobles o comillas simples, siempre que se usen las mismas para abrir y cerrar.
  - Diferencian entre mayúsculas y minúsculas.
  - El nombre del atributo debe comenzar por una letra o un guion bajo.
  - Pueden estar formados por caracteres alfanuméricos, guiones, guiones bajos y puntos.

```
<nombre grupo="DIO">Holy Diver</nombre>
```

Sintaxis. Relaciones entre elementos

- Las reglas para estructurar los elementos entre sí son las siguientes:
  - Existe un **único elemento raíz**.
  - Un elemento puede contener otros elementos o texto.
  - Cada uno de los elementos descendientes directos de un elemento se llama hijo (**child**).
  - El elemento ascendiente directo de un elemento se llama padre (**parent**).
  - Los elementos con un padre en común se denominan hermanos (**sibling**).
  - No se puede crear un elemento como hijo de otro y realizar el cierre después del cierre del elemento padre.

Comentarios

- XML permite incluir comentarios en el cuerpo del documento.
- Están formados por la apertura <!-- y el cierre -->
- Los comentarios no son procesados, ni validados, ni participan en la estructura del documento.

```
<!--Todo lo que incluya aquí es un comentario -->
```

Entidades

- Las entidades son un mecanismo para representar información dentro de un documento XML haciendo referencia a ella en lugar de tener que escribirla directamente.
- Esta información puede ser desde un carácter predefinido o bloques enteros de información.
- Las entidades las definimos en el DTD. Cuando creamos la entidad se asigna un “alias” a ese bloque de texto, que se podrá invocar desde el documento XML al ser procesado.

Entidades XML predefinidas

Símbolo a representar	**Entidad**
<	&lt;
>	&gt;
‘	&apos;
“	&quot;
&	&amp;

XML bien formado vs XML válido

- Un documento XML está **bien formado** cuando cumple con las **reglas generales** de creación de un documento XML (las reglas que hemos visto hasta ahora).
- Un documento XML es **válido** cuando, además de estar bien formado, cumple con las reglas que establecemos en su **DTD**.

CDATA

- Una sección CDATA en xml contiene un conjunto de caracteres que no será evaluado ni tratado por el analizador.
- Se usan frecuentemente para almacenar código que provocaría un problema con el resto del documento si se evaluase.
- <![CDATA[
-]]>
- La información contenida en CDATA no forma parte de la estructura jerárquica del documento, formando únicamente un elemento terminal.

Espacios de nombres

- En XML, los espacios de nombres (*namespaces*) son un mecanismo para **evitar conflictos de nombres** entre elementos y atributos de diferentes vocabularios o esquemas.
- El objetivo es evitar ambigüedades que podrían surgir en caso de que haya elementos o atributos con el mismo nombre.
- Esto ocurre cuando fusionamos varios ficheros XML en uno solo y se presentan elementos con el **mismo nombre**, pero con **diferente significado**.

Espacios de nombres

- Los *namespaces* asignan un **prefijo único** a un conjunto de elementos y atributos de un **vocabulario específico**, permitiendo que los elementos y atributos de diferentes vocabularios coexistan en el mismo documento sin causar conflictos.
- También es posible no indicar ningún prefijo. En ese caso, el *namespace* hace referencia a todos los elementos y atributos en el documento. Es decir, existe un **vocabulario único** para todo el documento XML.

Namespaces con prefijo

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

<miDocumentoXML
    xmlns:etiquetas="http://www.miPagina/especificacionHTML"
    xmlns:muebles="http://www.miPagina/especificacionMuebles">

    <etiquetas:table>
      <etiquetas:tr>
        <etiquetas:td>Apples</etiquetas:td>
        <etiquetas:td>Bananas</etiquetas:td>
      </etiquetas:tr>
    </etiquetas:table>

    <muebles:table>
      <muebles:name>African Coffee Table</muebles:name>
      <muebles:width>80</muebles:width>
      <muebles:length>120</muebles:length>
    </muebles:table>
</miDocumentoXML>
```

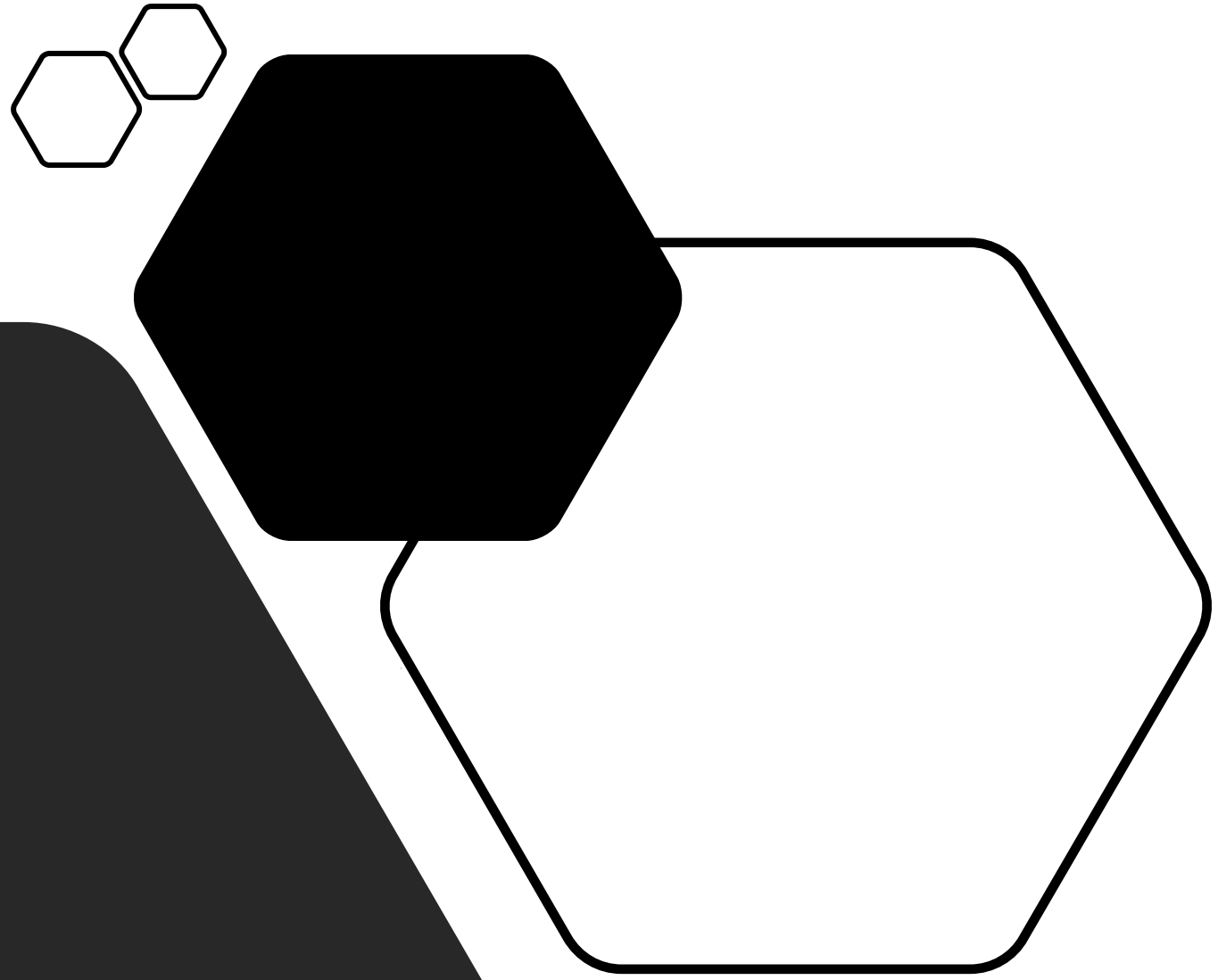
Namespaces sin prefijo

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

<miDocumentoXML>
    <table xmlns="http://www.miPagina/especificacionHTML">
      <tr>
        <td>Apples</td>
        <td>Bananas</td>
      </tr>
    </table>

    <table xmlns="http://www.miPagina/especificacionMuebles">
      <name>African Coffee Table</name>
      <width>80</width>
      <length>120</length>
    </table>
</miDocumentoXML>
```

DTD



DTD

- Un DTD contiene una serie de reglas que sirven para validar un XML.
- No es obligatorio tener un DTD para trabajar con XML, pero sí es recomendable.
- Las reglas contenidas en un DTD establecen la estructura del XML, así como los elementos y atributos permitidos.
- Es relativamente sencillo usar DTD, pero tiene algunas limitaciones, que se solucionan al usar otras tecnologías.

DTD interno y externo

- De forma similar a como usábamos CSS, podemos crear el DTD dentro del propio documento XML (DTD interno) o bien crear un archivo DTD que podemos vincular desde varios XML (DTD externo).
- Para crear un DTD interno, debemos declarar una sección DOCTYPE justo al terminar la cabecera XML:

```
<!DOCTYPE nombre_etiqueta_raíz[
      Reglas DTD
]>
```

DTD interno y externo

- Para crear un DTD externo debemos incluir el siguiente código en el XML, justo después de la cabecera:

```
<!DOCTYPE nombre_etiqueta_raíz SYSTEM "archivoDTD.dtd">
```

- En el archivo externo DTD escribimos directamente las reglas DTD.

Definición de entidades en DTD

- Ya hemos visto el funcionamiento de las entidades internas predefinidas en XML. Con DTD podemos definir nuestras propias entidades. La sintaxis es la siguiente:
- Entidades internas:
  - `<!ENTITY nombre_entidad "valor de la entidad">`
  - En el XML que use la entidad se referencia como &nombre_entidad;
  - Si abrimos el XML con un navegador, se sustituirá el nombre de la entidad con su valor.

Definición de entidades en DTD

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE clases[
    <!ENTITY grupo "1º DAM">
]>
<clases>
    <clase>
        <nombre>Lenguajes de marcas</nombre>
        <aula>112</aula>
        <grupo>&grupo;</grupo>
    </clase>
    <clase>
        <nombre>Base de datos</nombre>
        <aula>112</aula>
        <grupo>&grupo;</grupo>
    </clase>
</clases>
```

```
▼<clases>
 ▼<clase>
    <nombre>Lenguajes de marcas</nombre>
    <aula>112</aula>
    <grupo>1º DAM</grupo>
  </clase>
 ▼<clase>
    <nombre>Base de datos</nombre>
    <aula>112</aula>
    <grupo>1º DAM</grupo>
  </clase>
 </clases>
```

Definición de entidades en DTD

- Las entidades externas se definen fuera del DTD y se dividen en públicas y privadas, en función del uso que se le vayan a dar. Generalmente las que declaremos serán privadas.
- Declaración pública:

```
<!ENTITY nombre_entidad PUBLIC "id_publico" "ubicación">
```

- Declaración privada:

```
<!ENTITY nombre_entidad SYSTEM "ubicación">
```

Definición de elementos en DTD

- Cuando incluimos la definición de un elemento en un DTD, estamos definiendo cómo se puede usar ese elemento en los ficheros XML que validan contra ese DTD.
- Por ejemplo, si en el DTD incluimos lo siguiente:

```
<!ELEMENT nombre (#PCDATA)>
```

- Queremos decir que si usamos en el XML la etiqueta <nombre>, va a tener que contener forzosamente un texto. En general, la sintaxis es:

```
<!ELEMENT nombre contenido>
```

Definición de elementos en DTD

- Existen los siguientes tipos de contenido que podemos usar en la definición de elementos:
- EMPTY: indica que debe estar vacío.
- ANY: indica que puede contener cualquier contenido.
- (nombreElemento): indica que puede contener este elemento.
- (nombreElemento1, nombreElemento2, …. nombreElementoN): indica que debe contener la secuencia de elementos indicada.
- (#PCDATA): indica que puede contener texto.

Cardinalidades de elementos en DTD

- Cuando un elemento puede contener otros elementos podemos indicar las posibles combinaciones de éstos que son admitidas:
  - (nombreElemento): una ocurrencia del elemento.
  - (nombreElemento?): 0 o 1 ocurrencias del elemento.
  - (nombreElemento*): 0 o varias ocurrencias del elemento.
  - (nombreElemento+): 1 o varias ocurrencias del elemento.
  - (nombreElemento1, nombreElemento2, nombreElemento3): debe contener todos los elementos indicados.
  - (nombreElemento1 | nombreElemento2 | nombreElemento3): debe contener un elemento de los indicados.
- La combinación de estas posibilidades nos permitirá crear las reglas para la validación de nuestro XML.

Atributos de elementos en DTD

- Los DTD también permiten incluir reglas y condiciones para determinar la validez de los atributos. Esto se declara a nivel de elemento.
- La sintaxis es la siguiente:

```
<!ATTLIST nombre_elem nombre_atrib tipo_atrib valor_atrib>
```

- Ejemplo:

```
<!ATTLIST clase grupo CDATA "Desconocido">
```

Atributos de elementos en DTD

- **Tipo_atrib puede ser uno de los siguientes:**
- **CDATA**: El valor es una cadena de caracteres.
- **(valor1 | valor2 | …. | valorN)**: El valor es una lista de posibles valores.
- **ID**: El valor es un identificador único.
- **IDREF**: El valor es un id de otro elemento.
- **IDREFS**: El valor es una lista de ids de otros elementos, separadas por espacios.
- **NMTOKEN**: El valor debe cumplir las normas de nombre válido XML.
- **NMTOKENS**: El valor debe cumplir las normas de nombre válido XML.
- **ENTITY**: El valor es una entidad.
- **ENTITIES**: El valor es una lista de entidades.

Atributos de elementos en DTD

- **Valor_atrib puede ser uno de los siguientes:**
- **valor**: El valor asignado por defecto.
- **#REQUIRED**: Es obligatorio escribir el atributo.
- **#IMPLIED**: El valor es opcional.
- **#FIXED valor**: El atributo siempre tendrá el valor indicado.

XML Schema

XML Schema

- **XML Schema** es el nombre por el que se conoce a un lenguaje usado para describir la estructura y restricciones de un documento XML.
- Su nombre técnico es **XML Schema Definition** (XSD), aunque muchas veces se hace referencia a él únicamente como XML Schema (XS).
- XSD permite conseguir más precisión en el establecimiento de reglas de validación, superando las carencias de DTD.
- Su primera versión data de 2001, como alternativa a DTD.

XML Schema

- **Algunas de las ventajas que aporta XSD son:**
  - Determinar los elementos y atributos que admite un documento XML.
  - Permite determinar los tipos de datos admitidos por los elementos y los atributos.
  - Permite asignar valores por defecto, tanto a elementos como a atributos.
  - Permite determinar las cardinalidades de elementos de forma precisa.
  - Permite determinar las relaciones entre los elementos que forman el XML (hijos y padres, en número y orden).
- La extensión de un documento XSD es .xsd.

XML Schema: Declaración

- Para realizar una validación de un documento XML utilizando un XSD, debemos realizar dos pasos:
  - **Crear el XSD** y definir la estructura a validar en él.
  - **Vincular el XSD** creado en el **documento XML** para que se pueda utilizar para su validación.
- **Creación del XSD:**
  - La creación de un XSD se conoce como creación de un esquema. Los elementos XML que se usan para generar un esquema tienen que pertenecer al *namespace* XML Schema, que establece el prefijo xs: para todos ellos.

XML Schema: Declaración

- Un fichero XSD base tiene la siguiente estructura:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- Resto de elementos -->
</xs:schema>
```

- Siendo la declaración de la cabecera opcional, pero recomendable.
- El elemento *schema* siempre será el elemento raíz.

XML Schema: Declaración

- Dentro de *schema* se incluirán todos los elementos y atributos que puedan aparecer en un documento XML que deseemos validar.
- Algunos elementos hijos posibles son:
  - element
  - atribute
  - simpleType
  - complexType
  - group
  - attributeGroup

Vinculación de XML Schema

- Una vez que creado el XSD, el paso siguiente es vincularlo en el documento XML a validar.
- Para esto añadimos los siguientes atributos en la etiqueta raíz del documento XML:

```
<elemento_raiz
   xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
   xs:noNamespaceSchemaLocation="esquema.xsd">
   <!-- Resto de elementos del XML -->
</elemento_raiz>
```

Vinculación de XML Schema

- xmlns:xs debe contener siempre la URL "http://www.w3.org/2001/XMLSchema-instance", mientras que xs:noNamespaceSchemaLocation contiene la ruta hasta el archivo .xsd que queramos utilizar.
- La URL debe indicarse siempre con el protocolo http, ya que existen aplicaciones que no validan correctamente si usamos https.

XML Schema: Ejemplo

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

<famoso xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="famoso.xsd">
    Michael Jackson
</famoso>
```

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="famoso" type="xs:string"/>
</xs:schema>
```

XML Schema: tipos de elementos

- Podemos hacer una primera distinción de los tipos de elementos que podemos definir con XSD en simples y complejos:
  - **Elementos simples:** es aquel que solo contiene texto, no pudiendo contener otros elementos **ni tener atributos.**
  - **Elementos complejos:** es aquel que contiene uno o más elementos y/o atributos.
- Por tanto, en el momento en que un elemento tenga un atributo, pasará a ser de tipo complejo:

```
<disco grupo="Daft Punk">Discovery</disco>
```

XML Schema: tipos de elementos

- Otra categorización de los elementos es:
  - **Elementos locales:** son declarados como hijos de elementos que no son el elemento raíz, y solo pueden usarse una vez.
  - **Elementos globales:** hijos del elemento raíz. Pueden ser reutilizados.

XML Schema: elementos simples

- La sintaxis básica de XSD para definir elementos es:

```
<xs:element name=""></xs:element>
```

- Siendo *element* la etiqueta que determina un elemento y *name* el atributo que contiene cómo debe llamarse este elemento.
- Ejemplos de definición de elementos simples:

```
<xs:element name="nombre" type="xs:string"></xs:element>
<xs:element name="edad" type="xs:integer"></xs:element>
```

XML Schema: elementos simples

- Además de name, otros atributos que se pueden usar dentro de element son:
  - **type:** tipo de dato.
  - **minOccurs:** número mínimo de ocurrencias (por defecto 1).
  - **maxOccurs:** número máximo de ocurrencias (por defecto 1). Si queremos que sea ilimitado debemos escribir el valor unbounded.

XML Schema: elementos complejos

- Los elementos de tipo complejo pueden actuar a modo de contenedor. Se declaran utilizando la etiqueta **complexType** y pueden:
  - Contener a otros elementos, pudiendo opcionalmente contener atributos.
  - Estar vacíos, pudiendo opcionalmente contener atributos.
  - Tener contenido mixto: otros elementos y texto, pudiendo opcionalmente contener atributos.

XML Schema: elementos complejos

- La sintaxis básica de XSD para definir elementos complejos es:

```
<xs:element name="">
   <xs:complexType>
   </xs:complexType>
</xs:element>
```

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

    <famoso>Michael Jackson</famoso>

    <famoso>Frank Sinatra</famoso>

</famosos>
```

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="famosos">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="famoso" type="xs:string"
maxOccurs="unbounded"></xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

XML Schema: elementos complejos

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

<famosos xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="famosos.xsd">
    <famoso apodo="El rey del pop">Michael Jackson</famoso>
    <famoso apodo="La voz">Frank Sinatra</famoso>
</famosos>
```

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="famosos">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="famoso" maxOccurs="unbounded">
                    <xs:complexType mixed="true">
                        <xs:attribute name="apodo"
type="xs:string"></xs:attribute>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

XML Schema: subelementos

- Los elementos pueden contener a otros elementos (subelementos).
- Existen **tres tipos de subelementos**:
  - **xs:sequence**: declara una secuencia de elementos obligatorios. Deben aparecer todos y en el mismo orden.
  - **xs:choice**: declara un conjunto de elementos, de los cuales solo debe aparecer uno.
  - **xs:all:** declara una secuencia de elementos opcionales, de los cuales, pueden aparecer todos, algunos o ninguno. Además, pueden aparecer en cualquier orden.

XML Schema: atributos

- La sintaxis básica de los atributos es la siguiente:

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

- Las reglas que se le aplican son las siguientes:
  - Tienen que aparecer a continuación de la declaración del elemento.
  - El atributo name es el nombre del atributo.
  - Con el atributo type definimos el tipo de dato del atributo. Por defecto es anySimpleType, es decir, cualquier tipo de datos simple.
  - Existe un atributo use, que indica la obligatoriedad del atributo:
    - **required:** obligatorio.
    - **optional:** opcional.
    - **prohibited:** no se permite usar el atributo en el elemento.

XML Schema: atributos

- El atributo **default** permite asignar un valor por defecto.
- No existe un orden de aplicación de los atributos dentro del elemento.
- No existe cardinalidad.
- Lógicamente, no pueden tener hijos.

XML Schema: tipos predefinidos

Tipo de dato	Descripción
xs:string	Cadena de caracteres
xs:integer	Número entero
xs:decimal	Número decimal (la parte decimal se separa con un .)
xs:boolean	Dato lógico. Admite true y false.
xs:date	Fecha. El formato es AAAA-MM-DD
xs:time	Horas. Formato hh:mm:ss
xs:duration	Un periodo de tiempo, en formato PnYnMnDTnHnMnSn. Por ejemplo: P2Y6M5DT12H35M30S equivale a 2 años, 6 meses, 5 días, 12 horas, 35 minutos y 30 segundos

XML Schema: restricciones

- Con XSD podemos **restringir los posibles valores** que pueden tomar los atributos y elementos.
- A estas restricciones se las denomina **facetas**.
- Las restricciones se definen mediante el elemento ***restriction***:

```
<xs:restriction base="xs:integer">
    <!-- Restricciones -->
</xs:restriction>
```

- Siendo el atributo base, el tipo de atributo sobre el que aplicaremos restricciones.

XML Schema: restricciones

- La idea es partir del atributo base e ir acotando los posibles valores que puede tomar, es decir, pasar de una validación más permisiva a otra menos permisiva.
- El elemento *restriction* debe ir **dentro de otros elementos XSD** y las restricciones se expresan como elementos hijo de éste:

```
<xs:restriction base="xs:integer">
  <xs:minInclusive value="10"/>
  <xs:maxInclusive value="30"/>
</xs:restriction>
```

XML Schema: restricciones predefinidas

Faceta	Descripción
xs:length	Establece una longitud fija
xs:minLength	Establece una longitud mínima
xs:maxLength	Establece una longitud máxima
xs:minExclusive	Establece que el valor tiene que ser mayor que el indicado
xs:minInclusive	Establece que el valor tiene que ser mayor o igual que el indicado
xs:maxExclusive	Establece que el valor tiene que ser menor que el indicado
xs:maxInclusive	Establece que el valor tiene que ser menor o igual que el indicado
xs:enumeration	Establece una lista de posibles valores.
xs:totalDigits	Establece el número de dígitos que puede tener un número
xs:fractionDigits	Establece el número de decimales que puede tener un número

XML Schema: restricciones predefinidas

Faceta	Descripción
xs:whiteSpace	Establece cómo tratar los espacios en blanco, las tabulaciones y los saltos de línea
xs:pattern	Permite establecer un patrón de caracteres permitidos

XML Schema: definición de tipos de datos

- Hasta ahora hemos visto que se pueden crear elementos simples y complejos.
- La sintaxis que hemos usado para definir un tipo de dato simple se ha basado en usar un tipo de datos predefinido.
- No obstante, también podemos crear nuestros propios elementos simples con tipos de datos personalizados.

```
<xs:element name="dia" type="dia-semana"/>
```

- En este ejemplo podemos observar que el tipo día-semana no está predefinido, por lo que es tarea del programador indicar en qué consiste ese tipo.

XML Schema: definición de tipos de datos

- **simpleType** es un elemento en XSD que nos permite definir un tipo de dato simple personalizado, a partir de:
  - Un tipo de dato primitivo.
  - Otro tipo de dato simpleType.
- Se usa en combinación con *restriction* para restringir los valores posibles de un elemento o atributo.
- Los tipos de datos personalizados se definen dentro de un elemento simpleType y se pueden reutilizar en varios elementos y atributos a lo largo de un esquema, siempre que se le asigne un valor al atributo *name*.

```
<xs:simpleType name="tipo-simple">
  <!-- Definición del tipo simple -->
</xs:simpleType>
```

```
<xs:element name="dia_semana">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="lunes"/>
            <xs:enumeration value="martes"/>
            <xs:enumeration value="miércoles"/>
            <xs:enumeration value="jueves"/>
            <xs:enumeration value="viernes"/>
            <xs:enumeration value="sábado"/>
            <xs:enumeration value="domingo"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

```
<xs:element type="dia_semana"/>

.........
<xs:simpleType name="dia_semana">
    <xs:restriction base="xs:string">
        <xs:enumeration value="lunes"/>
        <xs:enumeration value="martes"/>
        <xs:enumeration value="miércoles"/>
        <xs:enumeration value="jueves"/>
        <xs:enumeration value="viernes"/>
        <xs:enumeration value="sábado"/>
        <xs:enumeration value="domingo"/>
    </xs:restriction>
</xs:simpleType>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="famosos">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="famoso" maxOccurs="unbounded">
                    <xs:complexType mixed="true">
                        <xs:attribute name="apodo" type="xs:string"/>
                        <xs:attribute name="fallecido">
                            <xs:simpleType>
                                <xs:restriction base="xs:string">
                                    <xs:enumeration value="si"/>
                                    <xs:enumeration value="no"/>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:attribute>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="famosos">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="famoso" type="famoso" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="famoso" mixed="true">
        <xs:attribute name="apodo" type="xs:string"/>
        <xs:attribute name="fallecido">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="si"/>
                    <xs:enumeration value="no"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
</xs:schema>
```

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="semana">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="dia_semana" type="dia_semana_num_anho" maxOccurs="7"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

    <xs:simpleType name="dia_semana">
        <xs:restriction base="xs:string">
            <xs:enumeration value="lunes" />
            <xs:enumeration value="martes" />
            <xs:enumeration value="miércoles" />
            <xs:enumeration value="jueves" />
            <xs:enumeration value="viernes" />
            <xs:enumeration value="sábado" />
            <xs:enumeration value="domingo" />
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="dia_semana_num_anho">
        <xs:simpleContent>
            <xs:extension base="dia_semana">
                <xs:attribute name="num_semana" type="xs:integer" use="required"/>
        </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

</xs:schema>
```