Tema 3: DTD
Definición de Tipo de Documentos

- 1º FPIGS Desarrollo de Aplicaciones Web
- Lenguajes de Marcas y Sistemas de Gestión de la Información

Sintaxis

```
<!DOCTYPE element DTD identifier
[
   declaration1
   declaration2
   ........
]>
```

- **DTD** empieza con el delimitador <!DOCTYPE
- Un **elemento** ordena al analizador diseccionar el documento desde el elemento raíz especificado.
- **Identificador DTD** es un identificador para la definición del tipo de documento, el cual puede ser la ruta hacia un archivo en el sistema o una dirección URL hacia un archivo en Internet.
- Los **corchetes** [ ] encierran una lista opcional de declaraciones de entidad llamada subconjunto interno.

DTD Interno

Un DTD se denomina DTD interno cuando los elementos se han declarado dentro del archivo XML. Para referenciarlo como DTD interno, el atributo ***standalone*** en la declaración XML se debe marcar con un ***yes***. Esto significa que la declaración funciona al margen de la fuente externa.

Sintaxis

```
<!DOCTYPE root-element [element-declarations]>
```

donde ***root-element*** es el nombre del elemento raíz y ***element-declarations*** es donde se declaran los elementos.

Ejemplo

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE pelicula [
<!ELEMENT pelicula (titulo)>
<!ELEMENT titulo (#PCDATA)>
]>

<pelicula>
   <titulo>Titanic</titulo>
</pelicula>
```

DTD Externo

En los DTD externos, los elementos se declaran fuera del archivo XML. Se puede acceder a ellos especificando los atributos del sistema que pueden ser o un archivo legal **.dtd** o una dirección URL válida.

Para referenciarlo como DTD externo, el atributo ***standalone*** en la declaración XML se debe marcar con un ***no***. Esto significa, que la declaración incluye información de una fuente externa.

Sintaxis

```
<!DOCTYPE root-element SYSTEM "file-name">
```

donde ***file-name*** es el archivo con la extensión .dtd

Ejemplo

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE address SYSTEM "address.dtd">
<address>
  <name>Juan Tomas</name>
  <company>Microsoft</company>
  <phone>900123123</phone>
</address>
```

Contenido de address.dtd

```
<!ELEMENT address (name,company,phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
```

Tipos

Podemos referirnos a DTD externos usando:

Identificadores de sistema

Permite especificar la localización de un archivo externo que contiene declaraciones DTD. La sintaxis es la siguiente:

```
<!DOCTYPE name SYSTEM "address.dtd" [...]>
```

Como puede observar contiene la palabra clave **SYSTEM** y una referencia URI señalando la localización del documento.

3. Componentes

Un DTD contendrá básicamente declaraciones de los siguientes componentes XML:

- **Elementos**

```
<name>Juan Tomas</name>
```

- **Atributos**

```
<img src="flower.jpg"/>
```

- **Entidades**

```
<!ENTITY pi "3.141592">                    => &pi; (entidad interna)
<!ENTITY textFile SYSTEM "fichero.txt">    => &textFile;
<!ENTITY miURL SYSTEM "http://www.as.com"> => &miURL;
```

Elementos

Sintaxis

```
<!ELEMENT elementname (content)>
```

- **ELEMENT** Se usa una declaración como indicación para el analizador que va a definir un elemento.
- **elementname** es el nombre del elemento (también llamado identificador genérico) que se está definiendo.
- **content** define qué contenido, si hubiera alguno, puede ir en el elemento.

Tipos de contenidos de elemento

El contenido de la declaración de elementos en DTD se categoriza en:

- **Contenido vacío**
- **Contenido de elemento**
- **Contenido mixto**
- **Cualquier contenido**

Contenido vacío

```
<?xml version="1.0"?>
<!DOCTYPE hr[
   <!ELEMENT address EMPTY>
]>
<address />
```

Contenido de elemento

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE address [
   <!ELEMENT address (name,company,phone)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT company (#PCDATA)>
   <!ELEMENT phone (#PCDATA)>
]>
<address>
   <name>Juan Jose</name>
   <company>Microsoft</company>
   <phone>900 100 200</phone>
</address>
```

Lista de operadores

Operador	Sintaxis	Descripción	Example
+	<!ELEMENTO nombre del elemento (elemento secundario1+)>	Indica que el elemento secundario se puede dar una o más veces dentro del elemento principal.	<!ELEMENTO dirección (nombre+)> El nombre del *elemento secundario* se puede dar una o más veces dentro de el nombre de elemento *dirección*.
*	<!ELEMENTO nombre del elemento (elemento secundario1*)>	Indica que el elemento secundario se puede dar cero o más veces dentro del elemento principal.	<!ELEMENTO dirección (nombre*)> El nombre del *elemento secndario* se puede dar cero o más veces dentro del nombre del elemento *dirección*.
?	<!ELEMENTO nombre del elemento (elemento secundario1?)>	Indica que el elemento secundario se puede dar cero veces o una vez dentro del elemento principal.	<!ELEMENTO dirección (nombre?)> El nombre del *elemento secundario* se puede dar cero o una vez dentro del nombre del elemento *dirección*.
,	<!ELEMENTO nombre del elemento (elemento secundario1, elemento secundario2)>	Da una secuencia de los elementos secundarios separados por comas, los cuales deben ser incluidos en el nombre del elemento.	<!ELEMENTO dirección (nombre, compañía)> La secuencia de elementos secundarios *nombre*, *compañía*, que puede darse en el mismo orden dentro del nombre del elemento *dirección*.
|	<!ELEMENTO nombre del elemento (elemento secundario1 | elemento secundario2)>	Permite hacer elecciones en el elemento secundario.	<!ELEMENTO dirección (nombre | compañía)> Le permite escoger entre los elementos secundarios *nombre* o *compañía*, que pueden darse dentro del nombre del elemento *dirección*.

Normas de sintaxis

Necesitamos seguir ciertas reglas si hay más de un contenido de elemento:

- **Secuencias**:

```
<!ELEMENT address (name,company,phone)>
```

- **Elecciones**:

```
<!ELEMENT address (mobile | landline)>
```

Contenido mixto

```
<!ELEMENT elementname (#PCDATA|child1|child2)*>
```

- **PCDATA** es el texto no revisado. #PCDATA debe ir al inicio en la declaración de contenido mixto.
- **child1**, **child2**.. son los elementos y cada uno de ellos debe tener su propia definición en la DTD.
- El operador (*) debe seguir el contenido de declaración mixta si los elementos secundarios se incluyen
- El (#PCDATA) y sus declaraciones de elementos secundarios deben ser separados por(|).

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE address [
   <!ELEMENT address (#PCDATA|name)*>
   <!ELEMENT name (#PCDATA)>
]>
<address>
   Aqui va texto mezclado con el elemento hijo.
   <name>Juan Jose</name>
</address>
```

Cualquier contenido

Se declara usando la palabra clave **ANY**. Mayormente se refiere a un elemento de categoría mixta.

ANY resulta útil cuando aun se tienen que decidir los contenidos que se permitirán en el elemento.

ANY indica que el texto (PCDATA) y/o cualquier elemento declarado en la DTD puede usarse en el contenido del ***elementname***.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE address [
   <!ELEMENT address ANY>
]>
<address>
  Aqui un poco de texto de ejemplo
  <street>calle principal</street>
  <trabajo/>
</address>
```

Atributos

Sintaxis:

```
<!ATTLIST element-name attribute-name attribute-type attribute-value>
```

- Empiezan con la palabra clave **<!ATTLIST** si el elemento contiene atributos.
- **element-name** especifica el nombre del elemento al que se aplica el atributo.
- **attribute-name** especifica el nombre del atributo que se incluye con el nombre del elemento.
- **attribute-type** define el tipo de atributo.
- **attribute-value** toma un valor fijo que el atributo debe definir.

Atributos - Ejemplo

```
<?xml version = "1.0"?>
<!DOCTYPE address [
<!ELEMENT address (name)>
<!ELEMENT name (#PCDATA)>
<!ATTLIST name id CDATA #REQUIRED>
 ]>
<address>
   <name id="123">Jose Tomas</name>
</address>
```

- Empieza con la declaración XML versión
- Sigue con la declaración del tipo de documento:

  `<!DOCTYPE address`
- En el cuerpo del DTD se han declarado elementos y atributos:

  `<!ELEMENT address (name)>`

  `<!ELEMENT name (#PCDATA)>`
- El atributo **id** para el elemento **name** es CDATA (datos formados por caracteres) y su valor es REQUERIDO.

  `<!ATTLIST name id CDATA #REQUIRED>`

Atributos - Tipos

Tipo	Descripción
Atributos de cadena	
CDATA	(Character Data) Es un texto. Podrán tener cualquier carácter.
Atributos de caso	
ID	Es un identificador que permite identificar al elemento de manera única en todo un documento XML. No debe aparecer más de una vez.
IDREF	Se usa para hacer referencia a la identidad de otro elemento. Se utiliza para establecer conexiones entre los elementos.
IDREFS	Se usa para referenciar múltiples identidades.
ENTITY	Representa un entidad externa declarada anteriormente en el documento.
ENTITIES	Representa una lista de entidades externas en el documento.

Atributos - Tipos

Tipo	Descripción
Atributos de caso	
NMTOKEN	Similar al CDATA, el valor del atributo consiste de un nombre XML válido (solo podrá tener letras, dígitos, guión "-", subrayado “_", punto "." y dos puntos ":“).
NMTOKENS	Similar al CDATA, el valor del atributo consiste en una lista nombres XML válidos.
Atributos enumerados	
NOTATION	Un elemento será referenciado a una anotación declarada en el documento DTD.
Enumeración	El valor puede ser uno de los pertenecientes a una lista de valores escritos entre paréntesis "( )" y separados por el carácter "|".

Valor del atributo

En cada declaración de atributo, se debe especificar cómo aparecerá el valor en el documento.

Se puede especificar si un atributo puede tener:

- **Un valor predeterminado**
- **Un valor fijo**
- **Se requiere**
- **Ser optativo**

Atributo – Valores predeterminados

Contiene un valor predeterminado. Los valores se pueden rodear con comillas independientes (') o dobles(")

```
<!ATTLIST element-name attribute-name attribute-type "default-value">
```

donde el valor predeterminado del atributo es definido.

En el ejemplo tenemos el elemento ***name*** con atributo ***id*** cuyo valor predeterminado es 0

```
<?xml version="1.0"?>
<!DOCTYPE address [
<!ELEMENT address (name)>
<!ELEMENT name (#PCDATA)>
<!ATTLIST name id CDATA "0">
 ]>
<address>
   <name id="123">
      Jose Tomas
   </name>
</address>
```

Atributo – Valores fijos

Se usa la palabra clave **#FIXED** seguida de un valor

```
<!ATTLIST element-name attribute-name attribute-type #FIXED "value">
```

```
<?xml version="1.0"?>
<!DOCTYPE address [
  <!ELEMENT address (company)*>
  <!ELEMENT company (#PCDATA)>
  <!ATTLIST company name NMTOKEN #FIXED “Microsoft">
]>
<address>
  <company name=“Microsoft">Aqui va un pequeño texto</company>
</address>
```

Atributo – Valores requeridos

Se usa la palabra clave **#REQUIRED**

```
<!ATTLIST element-name attribute-name attribute-type #REQUIRED>
```

En el ejemplo el atributo **id** es obligatorio.

```
<?xml version="1.0"?>
<!DOCTYPE address [
<!ELEMENT address (name)>
<!ELEMENT name (#PCDATA)>
<!ATTLIST name id CDATA #REQUIRED>
 ]>
<address>
   <name id="123">Jose Tomas</name>
</address>
```

Atributo – Valores optativos

Si el atributo no tiene un valor predeterminado, ni fijo, ni es requerido, entonces hay que declarar el atributo como **#IMPLIED**

```
<!ATTLIST element-name attribute-name attribute-type #IMPLIED>
```

En el ejemplo el atributo **id** es optativo.

```
<?xml version="1.0"?>
<!DOCTYPE address [
<!ELEMENT address (name)>
<!ELEMENT name (#PCDATA)>
<!ATTLIST name id CDATA #IMPLIED>
 ]>
<address>
   <name>Jose Tomas</name>
</address>
```

7. Entidades

Permiten definir valores en un documento o utilizar otras entidades predefinidas.

Las entidades pueden ser de cuatro tipos:

- **Entidades integradas**
- **Entidades de carácter**
- **Entidades generales**
- **Entidades de parámetro**

Declaración de entidad en el DTD:

```
<!ENTITY entity_name "entity_value">
```

- **entity_name** es el nombre de la entidad.
- **entity_value** contiene el valor para el nombre de la entidad.
- El valor de la entidad para la entidad interna se puede referenciar añadiendo el prefijo & al nombre de la entidad: *&entity_name;*

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE address [
<!ELEMENT address (#PCDATA)>
<!ENTITY name "Jose Tomas">
<!ENTITY company "Microsoft">
<!ENTITY phone "900100200">
]>
<address>
   &name;
   &company;
   &phone;
</address>
```

Entidad integrada: Ejemplo

```
<?xml version="1.0"?>
<note>
   <description>Simon &amp; Garfunkel</description>
</note>
```

El **&** es reemplazado por **&** cuando el procesador lo encuentre:

`Simon & Garfunkel`

Entidad	Carácter
	Espacio en blanco
&gt;	>
&lt;	<
&quot;	“
&apos;	‘
&amp;	&

Entidades de carácter

- Se usan para nombrar alguna de las entidades que son representaciones simbólicas de información

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE author[
<!ELEMENT author (#PCDATA)>
<!ENTITY writer "Jose Tomas">
<!ENTITY copyright "&#169;">
]>
<author>&writer;&copyright;</author>
```

Se ha usado © como valor para el carácter copyright ©

Entidades generales

```
<!ENTITY entity-name "text">
```

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ENTITY source-text “Lenguajes de marcas">
]>
<note>
&source-text;
</note>
```

Tema 3: XSD
Documentos de esquema XML

1º FPIGS Diseño de Aplicaciones Web

Lenguajes de Marcas y Sistemas de Gestión de la Información

Enlace de XML-XSD

Para enlazar un documento XML con el XSD que lo define se deben usar las siguientes etiquetas:

1. **En el XML**

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

2. **En el XSD**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="elemento-raiz">
       …
```

Elementos simples en XSD

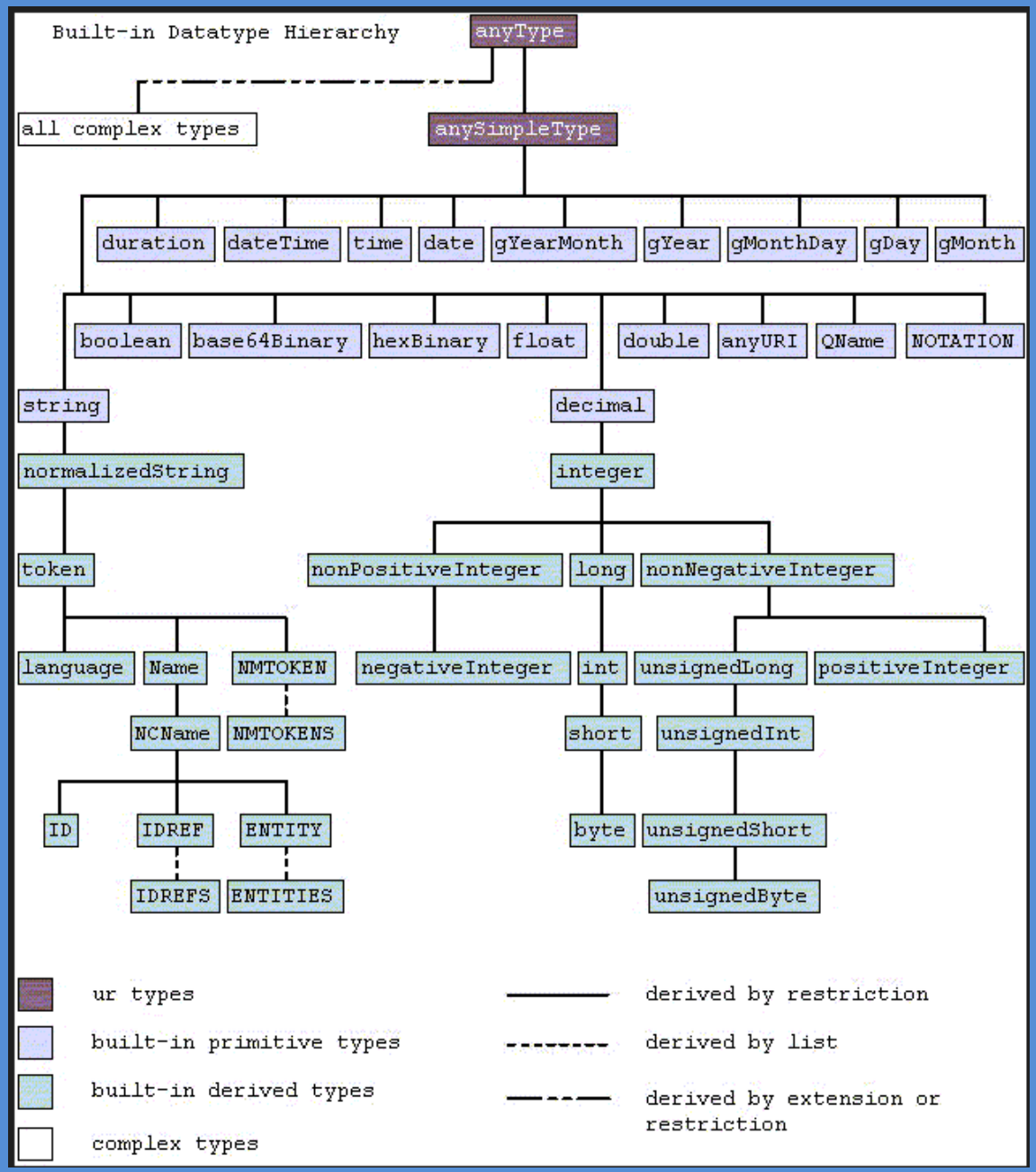
Los elementos simples solamente pueden contener texto (caracteres).

Los tipos de datos predefinidos pueden ser primitivos (**string**, **boolean**, **decimal**...) o derivados de estos (**integer**, **ID**, **IDREF**...)

Sintaxis:

```
<xs:element name="nombre_del_elemento" type="tipo_de_dato"/>
```

Jerarquía de tipos de datos en XSD:



EJEMPLO

Para los siguientes elementos XML:

```
<nombre>Elsa</nombre>

<edad>23</edad>
```

Sus definiciones pueden ser:

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

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

Tipos de declaración: fixed, default

Si se quiere indicar que un valor es fijo (fixed), se puede escribir, por ejemplo:

```
<xs:element name="mes" type="xs:string" fixed="agosto"/>
```

También, se puede especificar un valor por defecto (default), por ejemplo, tecleando:

```
<xs:element name="mes" type="xs:string" default="agosto"/>
```

Atributos en XSD

Sintaxis:

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

EJEMPLO: Para el elemento "curso" siguiente, donde aparece el atributo "grupo":

```
<curso grupo="B">2</curso>
```

Sus definiciones pueden ser:

```
<xs:element name="curso" type="xs:integer"/>

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

- Todos los atributos pueden tomar por valor tipos simples.

Tipos de atributos: fixed, default, optional, required

Ejemplo tipo **fixed:**

```
<xs:attribute name="grupo" type="xs:string" fixed="B"/>
```

Ejemplo tipo **default:**

```
<xs:attribute name="grupo" type="xs:string" default="B"/>
```

Ejemplo tipo **required:**

```
<xs:attribute name="grupo" type="xs:string" use="required"/>
```

Por defecto, si no se indica nada, el atributo será opcional (use="optional").

3. Restricciones (facetas) en XSD

XML Schema permite definir restricciones a los posibles valores de los tipos de datos. Ejemplos:

Faceta	*Descripción*
`Length`	Especifica una longitud fija.
`minLength`	Especifica una longitud mínima.
`maxLength`	Especifica una longitud máxima.
`Pattern`	Especifica un patrón de caracteres admitidos.
`Enumeration`	Especifica una lista de valores admitidos.
`whiteSpace`	Especifica cómo se debe tratar a los posibles espacios en blanco, las tabulaciones, los saltos de línea y los retornos de carro que puedan aparecer.
`maxInclusive`	Especifica que el valor debe ser menor o igual que el indicado.
`maxExclusive`	Especifica que el valor debe ser menor que el indicado.
`minExclusive`	Especifica que el valor debe ser mayor que el indicado.
`minInclusive`	Especifica que el valor debe ser mayor o igual que el indicado.
`totalDigits`	Especifica el número máximo de dígitos que puede tener un número.
`fractionDigits`	Especifica el número máximo de decimales que puede tener un número.

Ejemplo faceta

```
<xs:element name="mes">
   <xs:simpleType>
      <xs:restriction base="xs:integer">
         <xs:minInclusive value="1"/>
         <xs:maxInclusive value="12"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

- **xs:simpleType** permite definir un tipo simple y especificar sus restricciones.
- **xs:restriction** sirve para definir restricciones de un *xs:simpleType* (como se ha hecho en este ejemplo). También sirve para definir restricciones de un xs:simpleContent o de un xs:complexContent.
- **base** indica el tipo de dato a partir del cual se define la restricción.
- **xs:minInclusive** sirve para especificar que el valor debe ser mayor o igual que el indicado en su atributo value, (en este caso, mayor o igual que 1).
- **xs:maxInclusive** sirve para especificar que el valor debe ser menor o igual que el indicado en su atributo value, (en este caso, menor o igual que 12).

Ejemplo enumeration

Podría definirse como:

```
<xs:element name="color">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:enumeration value="verde"/>
         <xs:enumeration value="amarillo"/>
         <xs:enumeration value="rojo"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

- **xs:enumeration** sirve para definir una lista de valores admitidos. En nuestro ejemplo, color puede tomar los valores verde, amarillo o rojo.

Ejemplo pattern

```
<xs:element name="letra">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:pattern value="[a-z]"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

- **xs:pattern** sirve para definir un patrón de caracteres admitidos (en este caso se admite una única letra minúscula de la "a" a la "z"). El valor del patrón tiene que ser una expresión regular.

Patrón	Significado
[A-Z a-z]	Letra.
[A-Z]	Letra mayúscula.
[a-z]	Letra minúscula.
[0-9]	Dígitos decimales.
\D	Cualquier carácter excepto un dígito decimal.
(A)	Cadena que coincide con A.
A | B	Cadena que es igual a la cadena A o a la B.

Patrón	Significado
AB	Cadena que es la concatenación de las cadenas A y B.
A?	Cero o una vez la cadena A.
A+	Una o más veces la cadena A.
A*	Cero o más veces la cadena A.
[abcd]	Alguno de los caracteres que están entre corchetes.
[^abcd]	Cualquier carácter que no esté entre corchetes.

PATRÓN	DESCRIPCIÓN
[a-zA-Z0-9]	Cualquiera entre a-z, A-Z y 0-9 una vez
[a-z][a-z][0-5]	Dos letras minúsculas y un número entre 0 y 5
[a-z]{4}	Cuatro letras entre a y z
(palabra){2,5}	Entre 2 y 5 veces seguidas la cadena “palabra”
(palabra1|palabra2)+	Uno o más veces palabra1 o palabra2 juntas
c.....s	El . Significa cualquier carácter, por ejemplo aquí podría encajar “cocheras” o “c123456s”
[^cC][a-zA-Z]+	Palabra de dos caracteres que no empiece por “c” o “C”
\w[_@-](código)	\w coincide con [a-zA-Z0-9_]). En este caso por ejemplo 3@código sería válido
(li[^v]e)*(123)	Dos partes separadas entre paréntesis. En este ejemplo sería válido likelikelike123
(parte1)\.*(parte2)	parte1parte2 y parte1.....parte2 . No confundir \. con .
/[\w._%+-]+@[\w.-]+\.[a-zA-Z]{2,4}/	Correo electrónico

Elementos complejos en XSD

Un elemento es complejo (**complexType**) cuando contiene uno o más elementos y/o atributos.

Elemento vacío ejemplo

```
<xs:element name="bola">
  <xs:complexType>
    <xs:attribute name="numero" type="numeroDeBola"/>
  </xs:complexType>
</xs:element>

<xs:simpleType name="numeroDeBola">
   <xs:restriction base="xs:positiveInteger">
      <xs:minInclusive value="1"/>
      <xs:maxExclusive value="90"/>
   </xs:restriction>
</xs:simpleType>
```

- **xs:positiveInteger** indica que el valor del atributo numero debe ser un número entero mayor que cero.

En el ejemplo se ha definido vacío el elemento "bola", no pudiendo contener ni otros elementos ni texto. Ahora bien, véase que sí tiene un atributo, llamado numero.

Contenido mixto ejemplo

```
<xs:element name="persona">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="ciudad" type="xs:string"/>
      <xs:element name="edad" type="xs:positiveInteger"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

En el ejemplo se ha definido el elemento "persona" de tipo complejo mixto (mixed="true"):

Indicadores en XSD

- **Indicadores de orden**: secuencia (sequence), todo (all) y elección (choice).
- **Indicadores de ocurrencia**: maxOccurs y minOccurs.
- **Indicadores de grupo**: de elementos (group) y de atributos (attributeGroup).