

# ÍNDICE

## 1. DEFINICIÓN DE ESQUEMAS Y VOCABULARIOS -DTD

### 1. INTRODUCCIÓN

### 2. DTD

- 2.1. CÓMO AGREGAR UNA DTD A UN DOCUMENTO XML
- 2.2. ELEMENTOS
- 2.3. COMENTARIOS
- 2.4. ATRIBUTOS
- 2.5. ESPACIOS DE NOMBRES EN UNA DTD
- 2.6. DEFINICIÓN DE ENTIDADES DE DTD
- 2.7. DISEÑO Y VALIDACIÓN

### 1. INTRODUCCIÓN

- ❑ **XML** ha sido propuesto como estándar en el intercambio de la información, independiente de la plataforma en la que se genere o utilice. Esta premisa, es válida, mientras los documentos XML no cambien entre los intercambios.
- ❑ No sólo es imprescindible que el documento esté bien formado, sino que ambos actores (emisor y receptor) se ciñan a un formato de fichero definido previamente. Para evitar estos casos, se ha de definir una estructura fija del documento que conozcan las partes que intercambian la información.
- ❑ A continuación se mostrarán dos maneras de especificar formatos de comunicación en XML, de los que destacan las DTD (*Document Type Definition*) Y los XML Schema también llamados XSD (*XML Schema Definition*)

### 2. DTD

- ❑ **DTD** describe el lenguaje de marcado que se ha creado, es decir, define que el documento XML es válido si cumple las reglas de una determinada DTD.
- Una DTD indica:
  - Qué elementos pueden ser utilizados en un tipo de documento específico
  - Cuales son obligatorios y cuales opcionales
  - Cuales son repetibles y cuales no
  - En qué orden deben aparecer
  - Cómo deben anidarse los elementos que conforman un documento
- La DTD también contiene:
  - La declaración de las entidades que se utilizan en el documento:
    - Recursos externos XML
    - Recursos externos no XML: gráficos, multimedia, etc.
    - Texto que actúa como “comodín” o “abreviatura” para palabras de uso frecuente o términos que cambian con facilidad
  - La declaración de notaciones (instrucciones para procesar las entidades no XML)
- La DTD utiliza una sintaxis especial para definir la estructura de un tipo de documento. Esta sintaxis utiliza los siguientes elementos: `ELEMENT`, `ATTLIST`, `ENTITY`, `NOTATION` y `Comentarios`

### 2.1. CÓMO AGREGAR UNA DTD A UN DOCUMENTO XML

- ❑ Existen dos formas de agregar las DTD a los documentos XML: por medio de un vínculo externo o incrustando la DTD dentro del documento XML. Independientemente de la forma elegida para agregar una DTD, es necesario utilizar una **declaración de tipo de documento**.

```
<!DOCTYPE elemento_raiz ... >
```

- La sentencia `<!DOCTYPE>`, en primer lugar, indica al analizador sintáctico que se ha proporcionado una DTD para el documento y que tiene que leerla como parte del procesamiento del documento si se trata de un analizador sintáctico validador. La parte *elemento\_raiz de la* declaración de tipo de documento, indica el nombre del elemento raíz del documento para el que se proporciona la DTD.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
```

```
<!DOCTYPE LIBROS SYSTEM "libros.dtd">
```

```
<LIBROS>
```

```
...
```

```
</LIBROS>
```

- La declaración SYSTEM identifica esta DTD como parte de una selección local, o no pública de DTD. La referencia a la URL de una DTD puede ser relativa o absoluta. Si la dirección es **absoluta**, es necesario incluir la referencia *http:// completa del archivo en la* declaración de tipo de documento, por ejemplo:

```
<!DOCTYPE LIBROS SYSTEM "http://www.dtdfactory.com/libros.dtd" >
```

### 2.1. CÓMO AGREGAR UNA DTD A UN DOCUMENTO XML

- Una URL **relativa** puede incluir la parte del nombre de referencia del archivo que sea necesario. Por ejemplo, una URL relativa podría incluir una designación de directorio ("/dtds/libros.dtd") o simplemente la designación del archivo ("libros.dtd").
- Se puede utilizar a la vez una DTD interna y otra externa. Esto permite hacer uso de la DTD incrustada para los elementos específicos del documento y tener la posibilidad de hacer referencia a otros elementos desde una DTD estándar en otra ubicación.
- La declaración de tipo de documento se modifica ligeramente si se va a incrustar realmente su código en el documento XML, como se puede observar en el ejemplo siguiente:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE PFC [
  <!ELEMENT PFC ANY>
]>
<PFC>
<TITULO>SEWO: Sistema de Edición de Páginas Web On-Line</TITULO>
<CODIGO>ENI-9</CODIGO>
<ALUMNO>José Ramón Méndez Reboredo</ALUMNO>
<DIRECTOR>Florentino Fernández Riverola</DIRECTOR>
</PFC>
```

### 2.1. CÓMO AGREGAR UNA DTD A UN DOCUMENTO XML

- ❑ **DTD *PUBLIC* VS. *SYSTEM*:** Las DTD además de estar incrustadas en un documento o vinculadas al mismo, se pueden especificar como una definición de tipo de documento **PUBLIC** o **SYSTEM**. Independientemente del tipo de declaración es necesario especificar un nombre de archivo.
- En una declaración **PUBLIC** hay que incluir la URL completa del documento al que se hace referencia.

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
```

```
"http://java.sun.com/dtd/web-app_2_3.dtd">
```

- Las DTD privadas se marcan utilizando el identificador **SYSTEM**. Las DTD **SYSTEM** se encuentran habitualmente, en la misma ubicación o en el mismo sistema de red local que el propio documento.

```
<!DOCTYPE PFC SYSTEM "pfc.dtd">
```

### 2.2. ELEMENTOS

- ❑ **Elementos:** Los elementos se corresponden con los componentes estructurales de un documento, y definen su estructura lógica. Cada declaración de elemento se identifica por medio de la siguiente sintaxis básica:

```
<!ELEMENT nombre_elemento contenido>
```

- ✓ **nombre\_elemento:** Es el nombre del elemento, por ejemplo: TITULO, CODIGO o ALUMNO.
- ✓ **Contenido:** Cada elemento posee un conjunto de contenido permitido. La DTD identifica el tipo de contenido que se puede utilizar en un elemento en particular. Las opciones son “EMPTY”, “ANY”, contenido mixto (mixed content) o secundario (children). La siguiente tabla muestra las opciones de contenido válidas para elementos.

TIPO DE CONTENIDO	DESCRIPCIÓN	EJEMPLO
EMPTY	Especifica que el elemento puede no tener contenido, tanto si el contenido es texto o <u>elementos secundarios</u> .	<!ELEMENT IMAGE EMPTY>
ANY	Especifica que el elemento puede tener cualquier contenido, tanto si el contenido es texto, elementos secundarios o una <u>combinación de ambos</u> .	<!ELEMENT ADDRESSBOOK ANY>
Contenido mixto (mixed content)	Permite especificar el contenido exacto que se desee para el elemento. Se puede especificar sólo datos de texto (#PCDATA) o una combinación de texto y elementos secundarios específicos.	<!ELEMENT ADDRESSBOOK (NAME   NICKNAME   #PCDATA)>
Secundario (children)	Especifica qué elementos secundarios se pueden encontrar en el cuerpo del elemento identificado. El contenido no puede ser datos de caracteres.	<!ELEMENT CONTACT (NAME, STREET, CITY, PHONE)>

### 2.2. ELEMENTOS

- Considérese el siguiente ejemplo que trata de representar la información de la que se dispone acerca de los proyectos fin de carrera:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE PROYECTOS SYSTEM "proyectos01.dtd">
<PROYECTOS>
  <PFC>
    <TITULO>SEWO: Sistema de Edición de Páginas Web On-Line</TITULO>
    <CODIGO>ENI-9</CODIGO>
    <ALUMNO>José Ramón Méndez Reboredo</ALUMNO>
    <DIRECTOR>Florentino Fernández Riverola</DIRECTOR>
  </PFC>
</PROYECTOS>
```

- En este ejemplo aparece un elemento primario denominado PROYECTOS que sirve como elemento raíz del documento. Es necesario definir dicho elemento en la DTD antes de cualquier otro. La sentencia de declaración de elemento para PROYECTOS sería:

```
<!ELEMENT PROYECTOS (PFC) *>
```



### 2.2. ELEMENTOS

- Esta declaración de PROYECTOS indica que este elemento sólo puede contener elementos secundarios con el nombre PFC. El asterisco (\*) en el nombre PFC especifica que el elemento puede aparecer dentro de PROYECTOS cero o varias veces.
- La sentencia de declaración del elemento PFC aparecería como:  
`<!ELEMENT PFC (TITULO, CODIGO, ALUMNO, DIRECTOR)>`
- En este caso el elemento PFC sólo puede contener los elementos: TITULO, CODIGO, ALUMNO y DIRECTOR. Es necesario a su vez, definir cada uno de estos elementos:  
`<!ELEMENT TITULO (#PCDATA)>`  
`<!ELEMENT CODIGO (#PCDATA)>`  
`<!ELEMENT ALUMNO (#PCDATA)>`  
`<!ELEMENT DIRECTOR (#PCDATA)>`
- Cada uno de estos elementos se ha definido de forma que puede contener datos de caracteres (#PCDATA), pero no elementos secundarios.

### 2.2. ELEMENTOS

- ❑ **Control del contenido de un elemento:** El modelo de contenido especificado en la sentencia `<!ELEMENT>` de la DTD controla el contenido permitido para cada elemento. Esto identifica el contenido válido del elemento. El contenido que se permite para un elemento se denomina **modelo de contenido** (*content model*). La siguiente tabla define las reglas que se utilizan en una DTD para imponer el orden y la posibilidad de contenido de los distintos elementos.

MARCADO	DEFINICIÓN	EJEMPLO
( ... )	Delimita un grupo.	
A   B	Ocorre A o B, pero no ambos. En este caso se trata de un registro PAGER o de un registro CELL, pero no ambos.	<code>&lt;!ELEMENT CONTACT (PAGER   CELL)&gt;</code>
A, B	Ocurren A y B, y en ese orden. En este caso, el registro CONTACT sólo puede tener seis elementos secundarios y un elemento de datos de caracteres, que deben aparecer todos y en el orden en que están listados.	<code>&lt;!ELEMENT CONTACT (NAME, STREET, SUITE, CITY, STATE, ZIP, #PCDATA)&gt;</code>
A?	A ocurre cero o una vez. En este caso, el elemento NAME tiene que ocurrir una vez o ninguna dentro del elemento CONTACT.	<code>&lt;!ELEMENT CONTACT (NAME?)&gt;</code>
A*	A ocurre cero o más veces. En este caso, el elemento STREET puede ocurrir cualquier número de veces dentro del elemento CONTACT o no ocurrir.	<code>&lt;!ELEMENT CONTACT (STREET*)&gt;</code>
A+	A ocurre una o varias veces. En este caso, ADDRESSBOOK debe tener al menos un registro CONTACT en su interior.	<code>&lt;!ELEMENT ADDRESSBOOK (CONTACT+)&gt;</code>

### 2.2. ELEMENTOS

- ❑ **Control del contenido de un elemento:** El modelo de contenido especificado en la sentencia `<!ELEMENT>` de la DTD controla el contenido permitido para cada elemento. Esto identifica el contenido válido del elemento. El contenido que se permite para un elemento se denomina **modelo de contenido** (*content model*). La siguiente tabla define las reglas que se utilizan en una DTD para imponer el orden y la posibilidad de contenido de los distintos elementos.
- Algunos elementos no tiene contenido ni elementos secundarios. Se les denomina **elementos vacíos**. Para este tipos de elementos, el modelo de contenido se declara utilizando la palabra clave `EMPTY`.
- Utilizando las diferentes reglas del modelo de contenido que se muestran en la tabla anterior, y haciendo uso de los elementos vacíos, se puede ampliar el ejemplo que trata de modelar la información disponible acerca de los proyectos fin de carrera. La DTD quedaría como sigue:

### 2.2. ELEMENTOS

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- El listado de proyectos se compone de varios proyectos individuales (al menos uno)-->
<!ELEMENT PROYECTOS (PFC)+>

<!-- Un proyecto concreto -->
<!ELEMENT PFC (DATOS-PROYECTO, DATOS-ALUMNO, DATOS-DIRECTOR, DATOS-CODIRECTOR?)>

<!-- Datos de identificación del proyecto -->
<!ELEMENT DATOS-PROYECTO (CODIGO, TITULO, PRESENTADO)>
<!ELEMENT CODIGO (#PCDATA)>
<!ELEMENT TITULO (#PCDATA)>
<!ELEMENT PRESENTADO EMPTY>

<!-- Datos del alumno que realiza el proyecto -->
<!ELEMENT DATOS-ALUMNO ((DNI | PASAPORTE), NOMBRE, E-MAIL+, TELEFONO*)>
<!ELEMENT DNI (#PCDATA)>
<!ELEMENT PASAPORTE (#PCDATA)>
<!ELEMENT NOMBRE (#PCDATA)>
<!ELEMENT E-MAIL (#PCDATA)>
<!ELEMENT TELEFONO (#PCDATA)>

<!-- Datos del profesor director del proyecto -->
<!ELEMENT DATOS-DIRECTOR (NOMBRE, DEPARTAMENTO, AREA)>
<!ELEMENT DEPARTAMENTO (#PCDATA)>
<!ELEMENT AREA (#PCDATA)>

<!-- Datos del codirector si procede-->
<!ELEMENT DATOS-CODIRECTOR (NOMBRE)>
```

### 2.2. ELEMENTOS

- Un documento XML compatible con esta definición de tipo de contenido podría ser el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE PROYECTOS SYSTEM "E:\Floro\Asignaturas\PSI\Ejemplos\proyectos02.dtd">
<PROYECTOS>
  <PFC>
    <DATOS-PROYECTO>
      <CODIGO>ENI-9</CODIGO>
      <TITULO>SEWO: Sistema de Edición de Páginas Web On-Line</TITULO>
      <!-- si el proyecto no se hubiera presentado, esta etiqueta podría omitirse -->
      <PRESENTADO/>
    </DATOS-PROYECTO>
    <DATOS-ALUMNO>
      <DNI>34.558.265-J</DNI>
      <NOMBRE>José Ramón Méndez Reboredo</NOMBRE>
      <E-MAIL>moncho@lsi2.ei.uvigo.es</E-MAIL>
      <E-MAIL>moncho@yahoo.com</E-MAIL>
      <!-- no tenemos información de ningún teléfono -->
    </DATOS-ALUMNO>
    <DATOS-DIRECTOR>
      <NOMBRE>Florentino Fernández Riverola</NOMBRE>
      <DEPARTAMENTO>Informática</DEPARTAMENTO>
      <AREA>Lenguajes y Sistemas Informáticos</AREA>
    </DATOS-DIRECTOR>
    <!-- el proyecto no tiene ningún codirector -->
  </PFC>
</PROYECTOS>
```

### 2.3. COMENTARIOS

- ❑ **Comentarios:** Los comentarios se utilizan para proporcionar instrucciones dentro de una DTD de XML al programador y a cualquiera que realice labores de mantenimiento. El agente de usuario (software que esté utilizando el usuario final) que ve el documento no los interpreta.
- Habitualmente se utilizan comentarios en las siguientes situaciones:
  - Para agregar comentarios a las secciones específicas de código que aparecen en una DTD.
  - Para ocultar código que aún está en fase de desarrollo.
  - Para tomar nota de modificaciones realizadas en versiones anteriores de un documento.
  - Para tomar nota del autor y de la fecha de edición del documento.
- Los comentarios pueden ocupar una única línea o varias líneas. Cualquier texto que aparezca entre los marcadores de comentario, no será visible a un visitante del sitio cuando se ejecute con un analizador sintáctico o un navegador. No es conveniente colocar más de dos guiones seguidos dentro de un comentario. En algunos navegadores antiguos, el comentario podría aparecer truncado y mostrarse a los visitantes la parte del comentario a continuación de los guiones múltiples.
- Ejemplo:

```
<!ELEMENT graphic EMPTY>
<!--este elemento se usará para incluir gráficos-->
```

### 2.4. ATRIBUTOS

- ❑ **Atributos:** No proporcionan información a las personas que leen el documento, pero sí lo hacen para el SW encargado de mostrarlo. Los atributos se pueden utilizar como instrucciones para la aplicación de XML o como información adicional que se complementa al elemento que se está utilizando.
- Un elemento puede tener tantos atributos como sea necesario para realizar su cometido, pero un elemento sólo puede tener un atributo con un determinado nombre. No es posible definir un atributo global que sirva para todos los elementos del documento. La estructura es:

`<!ATTLIST nombre_elemento nombre_atributo tipo_de_atributo valor_predeterminado>`

```
<PFC idioma="gallego">
...
</PFC>
```

- Este atributo debería declararse en la DTD:

```
<ELEMENT PFC (DATOS-PROYECTO, DATOS-ALUMNO, DATOS-DIRECTOR, DATOS-CODIRECTOR?)>
<!ATTLIST PFC idioma CDATA "español">
```

- Se pueden declarar los atributos antes o después de los elementos. Es posible declarar atributos varias veces para el mismo elemento, pero sólo se utiliza la primera declaración del atributo. Si se desea, también se pueden declarar atributos para elementos que no existen. En muchos casos aparecerá el mismo atributo aplicado a varios elementos diferentes.

### 2.4. ATRIBUTOS

- Además de las diferentes reglas que gobiernan la utilización de elementos y el contenido de texto, también es posible especificar otros tipos de datos para su utilización con los atributos. Todos los **tipos de datos** que aparecen a continuación se utilizan en las definiciones de **atributos** de la DTD para identificar el tipo del valor del atributo.
  - CDATA
  - ENTITY, ENTITIES
  - Enumerados
  - ID
  - IDREF, IDREFS
  - NMTOKEN, NMTOKENS
  - NOTATION

```
<!-- Datos del alumno que realiza el proyecto -->
<!ELEMENT DATOS-ALUMNO ((DNI | PASAPORTE), NOMBRE, EMAIL+, TELEFONO*)>
<!ATTLIST DATOS-ALUMNO sexo CDATA "mujer">
...
<!-- Datos del profesor director del proyecto -->
<!ELEMENT DATOS-DIRECTOR (NOMBRE, DEPARTAMENTO, AREA)>
<!ATTLIST DATOS-DIRECTOR sexo CDATA "mujer">
```



### 2.4. ATRIBUTOS

- ❑ **Atributos CDATA** : Es el tipo de atributo más general, permitiendo que su valor contenga cualquier **cadena de caracteres** de texto. La cadena puede contener las letras y números estándar (A-Z, 1-0) y la mayor parte de los signos de puntuación. Los únicos cuatro signos de puntuación excluidos de los valores para atributos de este tipo son: <, >, & y ". Estos caracteres se pueden utilizar por medio de sus referencias a entidad. Los siguientes son algunos ejemplos de atributos CDATA:

```
<PROJECT id="0511-099"> </PROJECT>
<OBJECT width="15.5"/>
<VIDEO source="testingimage.gif" />
<IMAGE height="10"" />
```

- Cada uno de esos atributos se definiría en la DTD utilizando las siguientes sentencias <!ATTLIST>:  

```
<!ATTLIST PROJECT id CDATA "">
<!ATTLIST OBJECT width CDATA "1">
<!ATTLIST VIDEO source="testingimage.gif" />
<!ATTLIST IMAGE height CDATA "1">
```
- Se pueden incluir comillas dobles o sencillas dentro de un valor CDATA si el propio valor está delimitado con el tipo opuesto de comillas. Por ejemplo, se puede utilizar el valor de atributo width=" 8.5' " para establecer un ancho de 8.5 pies, o el inverso, height=' 4' '.

### 2.4. ATRIBUTOS

- ❑ **Atributos ENTITY o ENTITIES** : El tipo de atributo **ENTITY** permite **vincular un documento con entidades externas no analizables** que hagan referencia a archivos binarios. Este tipo de atributo sólo puede contener el nombre de una entidad definida en la DTD.
- El tipo de atributo **ENTITY** se utiliza, típicamente, para insertar un archivo de imagen o de vídeo en el documento XML. En la medida en que el navegador del documento XML pueda admitir el tipo de imagen o de vídeo, se puede utilizar una serie de sentencias `<!ELEMENT>`, `<!ATTLIST>` y `<!ENTITY>` para identificar la entidad en el documento en el que se inserta.

```
<!ELEMENT VIDEO EMPTY>
<!ATTLIST VIDEO source ENTITY #REQUIRED>
<!ENTITY VID SYSTEM "video.rm" NDATA RM>
```

- Una vez definidas las referencias de la DTD, se está en disposición de agregar el archivo de vídeo al documento XML en la ubicación deseada utilizando el siguiente comando:  
`<VIDEO source="VID" />`
- El tipo de atributo **ENTITIES** es el mismo que el tipo **ENTITY**, **pero permite hacer referencia a varias entidades** dentro del documento XML. Para utilizar el ejemplo anterior, bastaría con agregar opciones de vídeo adicionales a las declaraciones `<!ENTITY>` de la DTD.

### 2.4. ATRIBUTOS

```
<!ELEMENT VIDEO EMPTY>
<!ATTLIST VIDEO source ENTITIES #REQUIRED>
<ENTITY VID_SYSTEM "video.rm" NDATA RM>
<ENTITY VID_SM_SYSTEM "video_sm.rm" NDATA RM>
<ENTITY VID_FR_SYSTEM "video_fr.rm" NDATA RM>
```

- A continuación, existe la posibilidad de hacer referencia a los tres vídeos utilizando la siguiente declaración de elemento en el documento XML.

```
<VIDEO source="VID VID_SM VID_FR" />
```

- ❑ **Atributos Enumerados:** Los atributos enumerados no están especificados por ninguna palabra clave de XML, sino por una **lista de valores de texto** posibles para el atributo. Cada valor debe estar separado por una barra vertical, o símbolo de conexión ( | ), y tiene que seguir las reglas para la creación de nombres de XML válidos. El siguiente podría ser un ejemplo de lista de atributos para un elemento CIUDAD que sólo puede contener una serie de código postales locales.

```
<!ATTLIST CIUDAD zip (32001 | 32002 | 32003 | 32004)>
```

- Si la DTD tuviera esa especificación para el atributo *zip del elemento CIUDAD*, las siguientes designaciones de elemento serían válidas:

```
<CIUDAD zip="32001" />
<CIUDAD zip="32004" />
```

- Pero el siguiente elemento originaría un error de validación con un analizador sintáctico validador:

```
<CIUDAD zip="32005" />
```

### 2.4. ATRIBUTOS

- ❑ **Atributos ID:** Cuando se necesita utilizar un atributo **para identificar un elemento de forma única** en un documento XML hay que crear un atributo con el tipo ID. Este tipo permite utilizar cualquier nombre conforme a los convenios de nombres de XML, pero prohíbe la utilización del mismo nombre en varios elementos.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE PROJECT [
  <!ELEMENT PROJECT (CONTACT+)>
  <!ELEMENT CONTACT (#PCDATA)>
  <!ATTLIST PROJECT cod_id ID #REQUIRED>
  <!ATTLIST CONTACT cod_id ID #REQUIRED>
```

```
]>

<PROJECT cod_id="p01">
  <CONTACT cod_id="ffr">Florentino Fdez-Riverola</CONTACT>
  <CONTACT cod_id="fdg">Fernando Díaz Gómez</CONTACT>
</PROJECT>
```

- ❑ **Atributos IDREF o IDREFS:** Los tipos de atributo IDREF e IDREFS permiten **hacer referencia a un valor de tipo de atributo ID utilizado previamente**. En el ejemplo mostrado en la sección “Atributos ID”, se utilizaría el tipo IDREF para agregar un segundo atributo a los elementos de CONTACT y hacer referencia al proyecto o proyectos (si se utiliza IDREFS) de los que forma parte un contacto determinado.

### 2.4. ATRIBUTOS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE PROJECT [
  <!ELEMENT PROJECT (CONTACT+)>
  <!ELEMENT CONTACT (#PCDATA)>
  <!ATTLIST PROJECT pid ID #REQUIRED>
  <!ATTLIST CONTACT cid ID #REQUIRED>
  <!ATTLIST CONTACT proj_id IDREF "">
]>
<PROJECT pid="p01">
  <CONTACT cid="ffr" proj_id="p01">Florentino Fdez-Riverola</CONTACT>
  <CONTACT cid="fdg">Fernando Díaz Gómez</CONTACT>
</PROJECT>
```

- Si se deseara incluir todos los proyectos en que ha trabajado un individuo en un único registro de contactos, se utilizaría el tipo de atributo **IDREFS** en lugar de **IDREF** en la sentencia `<!ATTLIST>` de `proj_id`. Esto permitiría que la entrada de XML de contacto fuera:

```
<CONTACT cid="ffr" proj_id="p01 p03 p05">Florentino Fdez-Riverola</CONTACT>
```

- Se puede utilizar el tipo **IDREF** o **IDREFS** para establecer relaciones entre los distintos elementos de un documento.

### 2.4. ATRIBUTOS

- ❑ **Atributos NMTOKEN o NMTOKENS:** Los atributos `NMTOKEN` se utilizan para **limitar los valores de un atributo a nombres de XML bien formados**. Se trata de cadenas de texto que comienzan con una letra o el guión de subrayado (`_`), sólo contienen letras, números o el guión de subrayado y no tienen ningún espacio en blanco. Estos tipos de atributos funcionan bastante bien para limitar la estructura de la información que se puede utilizar como valor de atributos sin limitar su declaración real.
- ❑ **Atributos NOTATION:** El tipo de atributo `NOTATION` trabaja de forma similar al tipo de atributo `ENTITY`. Permite que el atributo realice una referencia al nombre de una notación ya declara o que va a serlo en el documento XML. Las notaciones, se utilizan para identificar el formato empleado con la información que no es XML, como archivos de vídeo, audio e imágenes. Se pueden usar **para especificar qué aplicación utilizar para trabajar con las entidades no analizables**.

```
<!ELEMENT VIDEO EMPTY>  
<!ATTLIST VIDEO software NOTATION (RM) #REQUIRED>  
<!NOTATION RM SYSTEM "realplayer.exe" >
```

- En el ejemplo anterior se identifica VIDEO como un elemento vacío con un atributo denominado software de tipo NOTATION. NOTATION hace referencia a la notación RM especificada en la declaración `<!NOTATION>`. El valor de atributo predeterminado `#REQUIRED` indica que hay que aplicar este atributo al elemento.

### 2.4. ATRIBUTOS

- También es posible proporcionar al software varios NOTATIONS entre los que poder elegir, tal y como se muestra en el ejemplo siguiente:

```
<!ELEMENT VIDEO EMPTY>
<!ATTLIST VIDEO software NOTATION (RM | QT | MMP) #REQUIRED>
<!NOTATION RM SYSTEM "realplayer.exe" >
<!NOTATION QT SYSTEM "quicktime.exe" >
<!NOTATION MMP SYSTEM "mediaplayer.exe" >
```

- **Asignación de valores predeterminados:** Se pueden utilizar tres valores predeterminados: #REQUIRED, #IMPLIED y #FIXED para imponer el valor del atributo de forma automática o para imponer la existencia del atributo en el documento XML. Como se ha comentado anteriormente, se puede proporcionar un valor predeterminado para un atributo agregando el valor, entre comillas, después de la designación del tipo de atributo.
  - ✓ **#REQUIRED:** Se utiliza para asegurarse de que en el documento XML se especifica el valor del atributo para todas las ocurrencias del elemento sobre el cual se define.
  - ✓ **#IMPLIED:** Son opcionales. Se pueden especificar explícitamente o ignorarse por completo. Se pueden utilizar para permitir, pero no imponer, la existencia de un fragmento concreto de información.
  - ✓ **#FIXED:** Cuando se utiliza esta palabra clave hay que establecer un valor predeterminado para el atributo, que es, además, el único valor disponible para ese atributo.

### 2.4. ATRIBUTOS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE PROJECT [
  <!ELEMENT PROJECT (CONTACT+)>
  <!ELEMENT CONTACT (#PCDATA)>
    <!ATTLIST PROJECT cod_id ID #REQUIRED>
    <!-- ATTLIST PROJECT consultoria CDATA #FIXED "Arthur&Andersen" -->
    <!ATTLIST CONTACT cod_id ID #REQUIRED>
]>

<PROJECT cod_id="p01">
  <CONTACT cod_id="ffr">Florentino Fdez-Riverola</CONTACT>
  <CONTACT cod_id="fdg">Fernando Díaz Gómez</CONTACT>
</PROJECT>
```

- Cuando se utiliza un atributo `#FIXED` en una DTD, obliga a que el valor del atributo coincida con el valor especificado, pues en caso contrario se generará un error sintáctico. Si el atributo no está presente físicamente en el documento XML, entonces se asume el valor especificado para ese atributo en la DTD. Al contrario que los atributos `#REQUIRED`, no es necesario especificar los atributos `#FIXED`.
- No se puede utilizar el valor `#FIXED` ni ningún tipo de atributo predeterminado con el tipo de atributo `ID`. `#FIXED` designa un único valor para un atributo negando, por tanto, la utilización de un identificador único.



### 2.4. ATRIBUTOS

- ❑ **Definición de varios atributos:** Existe una forma alternativa de proporcionar una lista de atributos para cada elemento. Este método simplemente ahorra algo de escritura y quizá consigue mejorar un poco la ordenación y la interpretación de los atributos para las personas.
- Este sistema utiliza una única sentencia `<!ATTLIST>` para cada elemento y especifica varios atributos dentro de esa sentencia. Se puede utilizar este sistema de declaración de atributos con cualquier combinación de valores de atributos, valores predeterminados y tipos.

```
<!DOCTYPE PROJECT [  
  <!ELEMENT PROJECT (CONTACT+)>  
  <!ELEMENT CONTACT (#PCDATA)>  
    <!ATTLIST PROJECT cod id ID #REQUIRED consultoria CDATA #FIXED "Arthur&Andersen">  
    <!ATTLIST CONTACT cod_id ID #REQUIRED>  
>]
```

- ❑ **Utilización de atributos predefinidos:** En XML existen dos atributos predefinidos. Aunque hay que definir los atributos, sería conveniente limitarse a utilizarlos de la forma para la que se supone que han sido creados: `xml:space` como descriptor para el tratamiento de espacios en blanco en un elemento y `xml:lang` como descriptor para el idioma utilizado para escribir el elemento.

```
<!ATTLIST CONTACT xml:space (default | preserve) "preserve">
```

```
<FRASE_DE_BIENVENIDA xml:lang="en">  
  Hello world!  
</FRASE_DE_BIENVENIDA >  
  
<FRASE_DE_BIENVENIDA xml:lang="sp">  
  ¡Hola mundo!  
</FRASE_DE_BIENVENIDA >
```

### 2.5. ESPACIOS DE NOMBRES EN UNA DTD

- ❑ Puesto que el procesador de XML trata la declaración *`xmlns:prefix`* como un atributo, es necesario definirla dentro de la DTD que utiliza el documento. Esto obliga a calificar todos los nombres de elemento que se encuentran dentro del marcado de XML con la referencia a su espacio de nombres.
- Un medio de definir el espacio de nombres consiste en utilizar la siguiente sentencia:

```
<!ATTLIST nombre_elemento xmlns:prefijo "url_del_espacio_de_nombres" #IMPLIED>
```

- Si se aplica este atributo al elemento raíz de un documento, entonces el espacio de nombres debería estar disponible para todos los elementos secundarios y atributos del elemento raíz.
- Puesto que una declaración de espacio de nombres se aplica al elemento en que se identifica y a todos los elementos, atributos y contenidos secundarios de dicho elemento, se pueden declarar varios espacios de nombres en una misma declaración de elemento. Esto hace que ambas declaraciones estén disponibles y sean válidas para cada elemento y atributo secundario.
- Además de utilizar varios espacios de nombres en un único documento XML para mezclar su estructura, se pueden utilizar varios espacios de nombres para combinar varias DTD en un formato y validar el contenido de forma simultánea. Para hacerlo es necesario modularizar la DTD y colocar las declaraciones de dichos módulos en el documento.

## 2.5. ESPACIOS DE NOMBRES EN UNA DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<ALUMNOS xmlns:cal="http://www.ei.uvigo.es/~riverola/calificacion"
xmlns:obs="http://www.ei.uvigo.es/~riverola/observacion" xmlns="GENERAL">
  <ALUMNO>
    <NOMBRE>Juan Luís Hernández Pérez</NOMBRE>
    <DNI>34.254.698-J</DNI>
    <cal:NOTA>APROBADO</cal:NOTA>
    <obs:NOTA>Ha superado la materia en primera convocatoria</obs:NOTA>
    <obs:NOTA>Su nota real fue 6.25</obs:NOTA>
  </ALUMNO>
</ALUMNOS>
```

### 2.6. DEFINICIÓN DE ENTIDADES DE DTD

- ❑ Las DTD, además de definiciones de tipo de documento, incluyen declaraciones de **entidades**. Las entidades se utilizan para definir conjuntos de información, proporcionando una forma abreviada de agregar grandes cantidades de datos, ya sean binarios o de texto, a un documento XML.
- Las entidades pueden ser cualquier cosa, aunque típicamente suele tratarse de texto o fragmentos de XML. El navegador de documentos XML puede procesar entidades formadas por texto o datos XML. En función del tipo de información que contiene, una entidad puede ser analizable o no analizable. Los datos de **entidades analizables** se componen de texto XML bien formado. Dicho texto se ajusta al estándar de XML y puede ser otro documento XML, un documento escrito en cualquiera de los lenguajes descendientes de XML o incluso un documento XSL. Las **entidades no analizables** contienen texto que no es XML (un mensaje de correo electrónico, imágenes, audio, etc.).
- Las entidades se pueden clasificar en **entidades internas** y **externas**, que a su vez se pueden utilizar como **entidades generales** o como **entidades parámetro**.
- ❑ **Entidades internas:** Permiten sustituir una cadena de texto por unos caracteres más fáciles de recordar y de teclear. Se declaran con la sintaxis:  

```
<!ENTITY nuevoprod "KTD-50-789890 A5">
```
- Se referencian en el documento escribiendo el identificador de la entidad entre los caracteres & y ; por ejemplo:  

```
<producto>&nuevoprod;</producto>
```

## 2.6. DEFINICIÓN DE ENTIDADES DE DTD

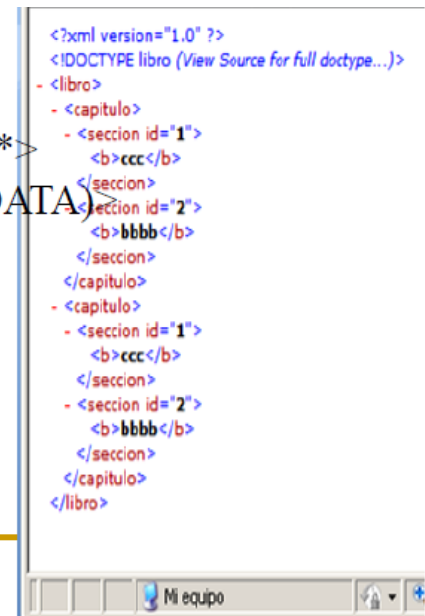
- ❑ **Entidades externas:** Referencian documentos XML externos a la entidad documento, que pueden verse como si se tratase de una única unidad. Permiten la reutilización, el trabajo en colaboración y la modularidad. Se declaran con la sintaxis:

```
<!ENTITY licencia SYSTEM "c:\licencia.xml">
```

- Se referencian con la sintaxis habitual: <docbody>&licencia;</docbody>

<pre>&lt;?xml version="1.0"?&gt; &lt;!DOCTYPE libro SYSTEM "libro.dtd"[ &lt;!ENTITY tema SYSTEM "tema.xml"&gt; ]&gt; &lt;libro&gt; &lt;capitulo&gt;&amp;tema;&lt;/capitulo&gt; &lt;capitulo&gt;&amp;tema;&lt;/capitulo&gt; &lt;/libro&gt;</pre>	<p>→</p>	<pre>&lt;?xml version="1.0"?&gt; &lt;!ELEMENT libro (capitulo)*&gt; &lt;!ELEMENT capitulo (#PCDATA)&gt;  &lt;seccion id="1"&gt; &lt;b&gt; ccc &lt;/b&gt; &lt;/seccion&gt; &lt;seccion id="2"&gt; &lt;b&gt; bbbb &lt;/b&gt; &lt;/seccion&gt;</pre>
---	----------	---

↘



### 2.6. DEFINICIÓN DE ENTIDADES DE DTD

- ❑ **Entidades externas no procesables:** Referencian cualquier archivo que no sea XML. Se declaran utilizando el calificador SYSTEM o PUBLIC, y van acompañadas de una notación (información adicional).

```
<!ENTITY logonscreenSYSTEM "c:\fm1.gif" NDATA gif>
```

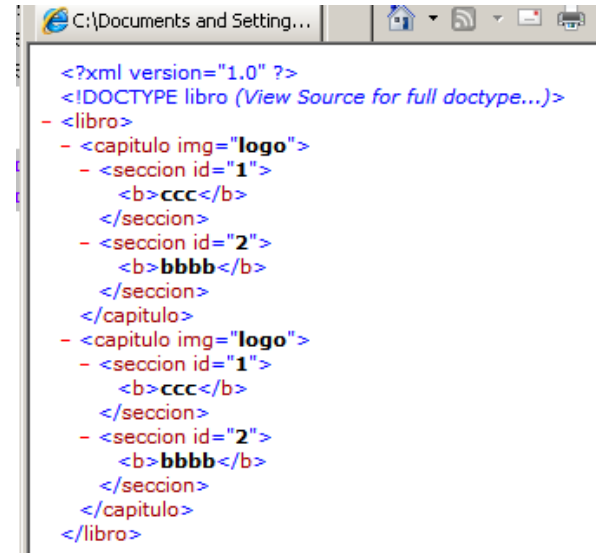
- La notación se escribe al comienzo de la DTD.

```
<!NOTATION gif SYSTEM "/programas/viewer.exe">
```

- Las notaciones pueden cumplir distintos propósitos:
  - Indicar el path del programa encargado de procesarla entidad (por ejemplo un visor especial)
  - Apuntar a un lugar en el que existe documentación sobre el formato, etc.
- La norma es abierta en este aspecto.

```
<?xmlversion="1.0"?><!DOCTYPE libro SYSTEM "libro.dtd" [  
<!NOTATION PNG SYSTEM "IExplore.exe">  
<!ENTITY tema SYSTEM "tema.xml">  
<!ENTITY logo SYSTEM  
"http://www.google.es/images/srpr/nav_logo37.png"  
NDATA PNG>  
<libro><capitulo img="logo">&tema;</capitulo>  
<capitulo img="logo">&tema;</capitulo>  
</libro>
```

```
.....  
<?xmlversion="1.0"?>  
<!ELEMENT libro (capitulo*,imagen)>  
<!ELEMENT capitulo (#PCDATA)>  
<!ATTLIST capitulo imgENTITY #REQUIRED>
```



```
<?xml version="1.0" ?>  
<!DOCTYPE libro (View Source for full doctype...)>  
- <libro>  
- <capitulo img="logo">  
- <seccion id="1">  
  <b>ccc</b>  
</seccion>  
- <seccion id="2">  
  <b>bbbb</b>  
</seccion>  
</capitulo>  
- <capitulo img="logo">  
- <seccion id="1">  
  <b>ccc</b>  
</seccion>  
- <seccion id="2">  
  <b>bbbb</b>  
</seccion>  
</capitulo>  
</libro>
```

### 2.6. DEFINICIÓN DE ENTIDADES DE DTD

- ❑ **Entidades parámetro internas y externas:** Se utilizan exclusivamente en la DTD (se declaran en la DTD al igual que las entidades normales, pero se les hace referencia sólo en la DTD).
  - Se declaran utilizando un carácter especial:  
`<!ENTITY % autorelem "nombre,apellido+">`
  - Para referenciarlas se escribe su nombre entre los caracteres % y ; , por ejemplo:  
`<!ELEMENT autores(noaut, %autorelem;)>`  
  
`.....`  
`<!ENTITY % p "subp, pp, foot">`  
`<!ELEMENT body (%p;)>`
  - De esta forma, la declaración de contenido del elemento body equivale a (subp, pp, foot)
  - El “modelo de contenido” al que sustituye la entidad se podrá reutilizar en otras partes de la DTD:  
`<!DOCTYPE texto [`  
`<!ENTITY % elemento-alf"<!ELEMENT ALF (#PCDATA)>">`  
`...`  
`%elemento-alf;`  
`]>`
  - También puede ser externa:  
`<!DOCTYPE texto [`  
`<!ENTITY % elemento-alf SYSTEM "alf.ent">`  
`...`  
`%elemento-alf;`  
`]`

### 2.7. DISEÑO Y VALIDACIÓN

- ❑ La DTD puede incluirse totalmente junto al documento XML (¡cuidado, en XML documento es la DTD y los datos!).
- ❑ La DTD puede estar definida en un archivo externo al documento al que se hará referencia desde este (en este caso, la DTD se podrá reutilizar y mantener con facilidad).
- ❑ La DTD puede estar definida en un documento externo, y también puede haber declaraciones en la entidad documento.

```
<?xml head version="1.0"?>  
<!ELEMENT head (body)>  
<!ELEMENT body  
  (#PCDATA)>
```

```
<?xml head version="1.0"?>  
<!DOCTYPE head SYSTEM "head.dtd">  
<head>  
<body>ejemplo</body>  
</head>
```

```
<?xml version="1.0"?>  
<!DOCTYPE head [  
  <!ELEMENT head (body)>  
  <!ELEMENT body (#PCDATA)>  
]>  
<head>  
<body>ejemplo</body>  
</head>
```

```
<?xml version="1.0"?>  
<!DOCTYPE head SYSTEM "head.dtd"  
  [  
    <!ENTITY car "coche">  
  ]>  
<head>  
<body>Ejemplo &car;</body>  
</head>
```



## 2.7. DISEÑO Y VALIDACIÓN

- ☐ Las DTDs externas son más fáciles de mantener, ya que los cambios se aplican automáticamente a todas sus instancias.
- ☐ Un documento será válido si cumple las restricciones que se indican en su DTD.
- ☐ Un documento será bien formado si los elementos están anidados correctamente, y si las entidades que referencia se han declarado.
- ☐ Un documento puede estar bien formado y ser no válido, ya que un documento XML puede no contener una declaración de tipo de documento.