



# XML Namespaces y DTD

Joan Sales  
IES Provençana  
DAWBIO1-M04-UF1

# Namespaces

```
<?xml version="1.0" encoding="UTF-8"?>
<prueba>
  <carta>
    <palo>Corazones</palo>
    <número>3</número>
  </carta>
  <carta>
    <plato>
      <nombre>Patatas fritas</nombre>
      <ingrediente>Patatas</ingrediente>
    </plato>
  </carta>
</prueba>
```

¿Tiene algún problema este documento .xml?

# Namespaces

```
<?xml version="1.0" encoding="UTF-8"?>
<prueba>
  <carta>
    <palo>Corazones</palo>
    <número>3</número>
  </carta>
  <carta>
    <plato>
      <nombre>Patatas fritas</nombre>
      <ingrediente>Patatas</ingrediente>
    </plato>
  </carta>
</prueba>
```

¿Tiene algún problema este documento .xml?

Esta well-formed... visualizarlo no da error...

# Namespaces

```
<?xml version="1.0" encoding="UTF-8"?>
<prueba>
  <carta>
    <palo>Corazones</palo>
    <número>3</número>
  </carta>
  <carta>
    <plato>
      <nombre>Patatas fritas</nombre>
      <ingrediente>Patatas</ingrediente>
    </plato>
  </carta>
</prueba>
```

¿Tiene algún problema este documento .xml?

Esta well-formed... visualizarlo no da error... Algunos programas pueden tener conflictos al leerlos, ya que hay etiquetas repetidas...

# Namespaces



Para resolver estos conflictos utilizamos prefijos.

```
<?xml version="1.0" encoding="UTF-8"?>
<prueba>
  <c:carta>
    <palo>Corazones</palo>
    <número>3</número>
  </c:carta>
  <m:carta>
    <plato>
      <nombre>Patatas fritas</nombre>
      <ingrediente>Patatas</ingrediente>
    </plato>
  </m:carta>
</prueba>
```

# Namespaces



Para resolver estos conflictos utilizamos prefijos. Estos prefijos hacen referencia a un namespace que se debe indicar con el atributo xmlns.

---

```
<?xml version="1.0" encoding="UTF-8"?>
<prueba>
  <c:carta xmlns:c="https://proven.cat/DAWBIO1/m05/poker">
    <palo>Corazones</palo>
    <número>3</número>
  </c:carta>
  <m:carta xmlns:f="https://proven.cat/DAWBIO1/m05/menu">
    <plato>
      <nombre>Patatas fritas</nombre>
      <ingrediente>Patatas</ingrediente>
    </plato>
  </m:carta>
</prueba>
```

# Namespaces



Sintaxis: `xmlns:prefijo="URI"`

---

```
<?xml version="1.0" encoding="UTF-8"?>
<prueba>
  <c:carta xmlns:c="https://proven.cat/DAW BIO1/m05/poker">
    <palo>Corazones</palo>
    <número>3</número>
  </c:carta>
  <m:carta xmlns:f="https://proven.cat/DAW BIO1/m05/menu">
    <plato>
      <nombre>Patatas fritas</nombre>
      <ingrediente>Patatas</ingrediente>
    </plato>
  </m:carta>
</prueba>
```

# Namespaces



Un Uniform Resource Identifier (URI) es un identificador único. No hace nada, solo es un nombre que nos permita diferenciar las etiquetas.

---

```
<?xml version="1.0" encoding="UTF-8"?>
<prueba>
  <c:carta xmlns:c="https://proven.cat/DAWBIO1/m05/poker">
    <palo>Corazones</palo>
    <número>3</número>
  </c:carta>
  <m:carta xmlns:f="https://proven.cat/DAWBIO1/m05/menu">
    <plato>
      <nombre>Patatas fritas</nombre>
      <ingrediente>Patatas</ingrediente>
    </plato>
  </m:carta>
</prueba>
```



# Namespaces



Nosotros definiremos los namespaces en el elemento root. De esta forma...

```
<?xml version="1.0" encoding="UTF-8"?>
<prueba xmlns:c="https://proven.cat/DAWBIO1/m05/carta" xmlns:f="https://proven.cat/DAWBIO1/m05/menu">
  <c:carta>
    <palo>Corazones</palo>
    <número>3</número>
  </c:carta>
  <m:carta>
    <plato>
      <nombre>Patatas fritas</nombre>
      <ingrediente>Patatas</ingrediente>
      <ingrediente>Cebolla</ingrediente>
      <receta>
        <paso>Las pides por aliexpress</paso>
        <paso>Te las comes</paso>
      </receta>
    </plato>
  </m:carta>
</prueba>
```

# Namespaces



O de esta otra...

```
<?xml version="1.0" encoding="UTF-8"?>
<prueba
  xmlns:c="https://proven.cat/DAWBIO1/m05/carta"
  xmlns:f="https://proven.cat/DAWBIO1/m05/menu">
  <c:carta>
    <palo>Corazones</palo>
    <número>3</número>
  </c:carta>
  <m:carta>
    <plato>
      <nombre>Patatas fritas</nombre>
      <ingrediente>Patatas</ingrediente>
      <ingrediente>Cebolla</ingrediente>
      <receta>
        <paso>Las pides por aliexpress</paso>
        <paso>Te las comes</paso>
      </receta>
    </plato>
  </m:carta>
</prueba>
```

# Namespaces

Importante: Todos los elementos sucesores pertenecen al mismo namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<prueba
  xmlns:c="https://proven.cat/DAWBIO1/m05/carta"
  xmlns:f="https://proven.cat/DAWBIO1/m05/menu">
  <c:carta>
    <palo>Corazones</palo>
    <número>3</número>
  </c:carta>
  <m:carta>
    <plato>
      <nombre>Patatas fritas</nombre>
      <ingrediente>Patatas</ingrediente>
      <ingrediente>Cebolla</ingrediente>
      <receta>
        <paso>Las pides por aliexpress</paso>
        <paso>Te las comes</paso>
      </receta>
    </plato>
  </m:carta>
</prueba>
```

Por ejemplo, palo y número pertenecen al namespace c.  
plato,nombre, ingrediente, receta, y paso pertenecen al namespace m

# Document Type Definition



Recordemos que no era lo mismo un documento well-formed que un documento válido.

Para que un documento .xml sea válido debe estar well-formed y además tener asociado un DTD.

Un DTD es otro archivo en que se especifica qué estructura debe seguir el documento .xml

Por estructura nos referimos a:

- Qué etiquetas tiene
- En qué orden deben estar esas etiquetas
- Qué atributos tiene cada etiqueta

# Document Type Definition

Hay dos maneras de especificar un DTD:

En el mismo documento

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE carta[
  <!ELEMENT carta (to,from,content)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT content (#PCDATA)>
]>

<carta>
  <to>me</to>
  <from>myself</from>
  <content>hi</content>
</carta>
```

En otro documento

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE carta SYSTEM "carta.dtd">

<carta>
  <to>me</to>
  <from>myself</from>
  <content>hi</content>
</carta>
```

Utilizaremos esta

# Document Type Definition



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE carta SYSTEM "carta.dtd">

<carta>
    <to>me</to>
    <from>myself</from>
    <content>hi</content>
</carta>
```

`<!DOCTYPE nombreElementoRoot SYSTEM path/archivo.dtd >`

# Document Type Definition



El archivo DTD contiene una lista de definiciones.

`<!ELEMENT name (tipo contenido)>`

*name* es el nombre de la etiqueta

*tipo contenido* indica qué hay dentro de esa etiqueta, puede ser:

- #PCDATA ->Significan datos
- EMPTY ->No hay nada, es una etiqueta sin contenido
- ANY ->Cualquier cosa. No filtra demasiado...
- Otros nombres de etiquetas

# Document Type Definition



`<!ELEMENT name(tipo contenido)>`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE carta SYSTEM "carta.dtd">
<carta>Hola</carta>
```

/home/alumne/XML/carta.xml

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



¡Los paréntesis son importantes!

/home/alumne/XML/carta.dtd



# Document Type Definition



Varios nombres entre comas indican que la etiqueta debe tener esas etiquetas en ese orden.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE clase SYSTEM "clase.dtd">
<clase>
  <número>45</número>
  <planta>3</planta>
  <profesor/>
</clase>
```

/home/alumne/XML/clase.xml

```
<!ELEMENT clase (número, planta, profesor) >
<!ELEMENT número (#PCDATA) >
<!ELEMENT planta (#PCDATA) >
<!ELEMENT profesor EMPTY>
```

/home/alumne/XML/clase.dtd

# Document Type Definition



Para indicar que un elemento como mínimo debe aparecer una vez (1-\*) utilizamos +

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plato SYSTEM "plato.dtd">
<plato>
    <ingrediente>Patatas</ingrediente>
    <ingrediente>Aceite</ingrediente>
</plato>
```

/home/alumne/XML/plato.xml

```
<!ELEMENT plato (ingrediente+) >
<!ELEMENT ingrediente (#PCDATA) >
```

/home/alumne/XML/plato.dtd

# Document Type Definition



Para indicar que un elemento puede aparecer ninguna o más veces (0-\*) utilizamos \*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE carta SYSTEM "carta.dtd">
<carta>
    <mensaje>Hola</mensaje>
</carta>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE carta SYSTEM "carta.dtd">
<carta>
</carta>
```

/home/alumne/XML/carta.xml

```
<!ELEMENT carta (mensaje*) >
<!ELEMENT mensaje (#PCDATA) >
```

/home/alumne/XML/carta.dtd

# Document Type Definition



Para indicar que un elemento puede aparecer o no (0-1) utilizamos ?

```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE clase SYSTEM "clase.dtd">  
<clase>  
    <número>45</número>  
    <planta>3</planta>  
    <profesor/>  
</clase>
```

/home/alumne/XML/clase.xml

```
<!ELEMENT clase (número,planta,profesor?) >  
<!ELEMENT número (#PCDATA) >  
<!ELEMENT planta (#PCDATA)>  
<!ELEMENT profesor EMPTY>
```

/home/alumne/XML/clase.dtd

# Document Type Definition



Para indicar que puede aparecer un elemento u otro podemos usar |


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE examen SYSTEM "examen.dtd">
<examen>
    <test/>
</examen>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE examen SYSTEM "examen.dtd">
<examen>
    <explicar/>
</examen>
```

/home/alumne/XML/examen.xml

```
<!ELEMENT examen (test|explicar) >
<!ELEMENT test EMPTY>
<!ELEMENT explicar EMPTY>
```

Tengo que definir  
todos los elementos



/home/alumne/XML/examen.dtd

# Document Type Definition

Para indicar que puede aparecer un elemento u otro podemos usar |

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE examen SYSTEM "examen.dtd">
<examen>
  <preguntas>15</preguntas>
  <test/>
</examen>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE examen SYSTEM "examen.dtd">
<examen>
  <preguntas>15</preguntas>
  <explicar/>
</examen>
```

/home/alumne/XML/examen.xml

```
<!ELEMENT examen (preguntas,(test|explicar)) >
<!ELEMENT preguntas (#PCDATA) >
<!ELEMENT test EMPTY>
<!ELEMENT explicar EMPTY>
```

Si hay elementos  
separados por  
comas, aquellos que  
vayan con una OR (!)  
deben ir entre  
paréntesis.

/home/alumne/XML/examen.dtd

# Document Type Definition



Para indicar que puede haber contenido intercalado utilizaremos el siguiente formato:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE carta SYSTEM "carta.dtd">
<carta>
  Hola, escribimos desde la empresa
  <empresa>BSC</empresa>. Los directores
  <nombre>Pepe</nombre> y <nombre>Susana</nombre> le
  dan la bienvenida.
</carta>
```

/home/alumne/XML/carta.xml

```
<!ELEMENT carta (#PCDATA|empresa|nombre)* >
<!ELEMENT empresa (#PCDATA) >
<!ELEMENT nombre (#PCDATA)>
```

/home/alumne/XML/carta.dtd

# Document Type Definition



¡Vigilad! No son lo mismo...

```
<!ELEMENT carta (empresa|nombre|ciudad)* >  
<!ELEMENT empresa (#PCDATA) >  
<!ELEMENT nombre (#PCDATA)>  
<!ELEMENT ciudad (#PCDATA)>
```

```
<!ELEMENT carta (empresa,nombre,ciudad)* >  
<!ELEMENT empresa (#PCDATA) >  
<!ELEMENT nombre (#PCDATA)>  
<!ELEMENT ciudad (#PCDATA)>
```



# Document Type Definition



Ejemplo pedidos:

Hagamos juntos el .dtd de este .xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- <!DOCTYPE pedidos SYSTEM "example_pedidos_dtd.dtd" -->
- <pedidos>
  - <pedido>
    <cod_pedido>1002</cod_pedido>
    <sucursal>Barcelona</sucursal>
    <direccion>C/ Caspe num 5</direccion>
    <nombre_trabajador>Juan Lopez</nombre_trabajador>
    <fecha_pedido>12/09/2019</fecha_pedido>
    <codigo_articulo>93</codigo_articulo>
    <unidades>6</unidades>
    <precio_unidad>12</precio_unidad>
  </pedido>
  - <pedido>
    <cod_pedido>1003</cod_pedido>
    <sucursal>Barcelona</sucursal>
    <direccion>C/ Caspe num 5</direccion>
    <nombre_trabajador>Anna Rodriguez</nombre_trabajador>
    <fecha_pedido>20/09/2019</fecha_pedido>
    <codigo_articulo>95</codigo_articulo>
    <unidades>1</unidades>
    <precio_unidad>102</precio_unidad>
  </pedido>
  - <pedido>
    <cod_pedido>1004</cod_pedido>
    <sucursal>Pamplona</sucursal>
    <direccion>C/gran vía 23</direccion>
    <nombre_trabajador>Luis Perez</nombre_trabajador>
    <fecha_pedido>21/09/2019</fecha_pedido>
    <codigo_articulo>94</codigo_articulo>
    <unidades>12</unidades>
    <precio_unidad>14</precio_unidad>
  </pedido>
  - <pedido>
    <cod_pedido>1005</cod_pedido>
    <sucursal>Madrid</sucursal>
    <direccion>C/Castellana 234</direccion>
    <nombre_trabajador>REbeca Mendez</nombre_trabajador>
    <fecha_pedido>22/09/2019</fecha_pedido>
    <codigo_articulo>94</codigo_articulo>
    <unidades>7</unidades>
    <precio_unidad>14</precio_unidad>
  </pedido>
</pedidos>
```

# Document Type Definition



## Atributos

`<!ATTLIST etiqueta nombre tipo valor>`

etiqueta -> Nombre de la etiqueta que contiene el atributo

nombre -> Nombre del atributo

tipo -> Varios disponibles, indica cómo son los datos del atributo

valor -> Indica si el atributo tiene un valor por defecto, si es opcional...

# Document Type Definition



Atributos. Tipos:

Type	Description
CDATA	Character Data (text that doesn't contain markup)
ENTITY	The name of an entity (which must be declared in the DTD)
ENTITIES	A list of entity names, separated by whitespaces. (All entities must be declared in the DTD)
<i>Enumerated</i>	A list of values. The value of the attribute must be one from this list.
ID	A unique ID or name. Must be a valid XML name.
IDREF	Represents the value of an ID attribute of another element.
IDREFS	Represents multiple IDs of elements, separated by whitespace.
NMTOKEN	A valid XML name.
NMTOKENS	A list of valid XML names, separated by whitespace.
NOTATION	A notation name (which must be declared in the DTD).

# Document Type Definition



Atributos. Tipos:

CDATA -> Texto plano, las etiquetas no se interpretan.

ENTITY -> Entidades declaradas en el DTD (una especie de alias, las veremos más adelante).

ENTITIES -> Varias entidades, separadas por espacios en blanco.

Enumerated (en1|en2|en3...) -> Varios valores especificados.

ID -> Un nombre único, no se puede repetir.

IDREF -> Hace referencia a un ID de otro elemento.

IDREFS -> Varios IDs de otros elementos, separados por espacios en blanco.

NMTOKEN -> Un nombre xml válido.

NMTOKENS -> Una lista de nombres xml válidos, separados por espacios en blanco.

# Document Type Definition



Atributos. Valores:

Puede ser:

Un valor directamente

#REQUIRED (Indica que el atributo es obligatorio)

#IMPLIED (El atributo es opcional)

#FIXED (El atributo solo puede tener el valor indicado)

<!ATTLIST cuenta ahorros CDATA "0">

<!ATTLIST cuenta ahorros CDATA #REQUIRED>

<!ATTLIST cuenta ahorros CDATA #IMPLIED>

<!ATTLIST cuenta ahorros CDATA #FIXED"100">

Si se especifica un valor directamente, aunque en un documento .xml no pongamos el atributo realmente estará, y tendrá el valor definido por defecto.

Tanto al indicar un valor directamente como al indicarlo con #FIXED se debe especificar dicho valor entre comillas

<!ATTLIST cuenta ahorros CDATA>

← Incorrecto.

# Document Type Definition



Atributos. Valores:

Es posible indicar que un atributo pueda tener determinados valores en concreto:

```
<!ATTLIST botella material (plástico|vidrio|metal) "plástico">
```

El valor por defecto debe ser uno de los permitidos.



```
<!ATTLIST botella material (plástico|vidrio|metal) #REQUIRED>
```

```
<!ATTLIST botella material (plástico|vidrio|metal) #IMPLIED>
```

# Document Type Definition



Atributos. Valores:

Podemos indicar varios atributos de dos formas:

Uno detrás de otro:

```
<!ATTLIST carta palo (tréboles|corazones|diamantes|picas) #REQUIRED número CDATA #REQUIRED>
```

O en líneas separadas

```
<!ATTLIST carta palo (tréboles|corazones|diamantes|picas) #REQUIRED>  
<!ATTLIST carta número CDATA #REQUIRED >
```

Mejor este...



# Document Type Definition



## Entities

Podemos definir entidades, una especie de alias en DTD:

```
<!ENTITY nombre "valor">
```

```
<!ENTITY copyright "powerpoint made by Joan for DAWBIO1M04 at the IES Provençana">
```

De esta forma podemos referirnos a esa entidad en el documento xml como si fuera un carácter especial.

```
<contenido> Documento bajo copyright: &copyright; </contenido>
```



