
M04 LLENGUATGE DE MARQUES

UF1. PROGRAMACIÓ AMB XML

A2: XML VÀLID

ÍNDEX

1. DTD (Document Type Definition)

- 1.1. Definició
- 1.2. Elements de cardinalitat
- 1.3. Tipus d'elements i atributs
- 1.4. Referència externa e interna a un DTD
- 1.5. Entitats

2. XSD (XML Schema Definition)

- 2.1. Definició
- 2.2. Elements simples
- 2.3. Elements complexos
- 2.4. Atributs
- 2.5. Restriccions
- 2.6. Validar un XML amb el seu corresponent Schema
- 2.7. Exemple XML Schema

1. DTD (Document Type Definition)

1.1. Definició.

Un DTD està format per una estructura que ens permet comprovar si un determinat document XML segueix les especificacions marcades per aquest DTD. En altres paraules, direm que un document XML és vàlid si segueix les restriccions definides en l'estructura del DTD perquè aquest actua com si fós una plantilla.

Un DTD bàsicament conté la definició de l'element arrel, la declaració dels elements intermedis tant simples com compostos i la definició dels atributs.

Exemple:

```
<!DOCTYPE clients [ Element arrel
<!ELEMENT clients (client+)>
<!ELEMENT client (IdClient, Empresa, Ciutat, Provincia)>
<!ELEMENT IdClient (#PCDATA)>
<!ELEMENT Empresa (#PCDATA)>
<!ELEMENT Ciutat (#PCDATA)>
<!ELEMENT Provincia (#PCDATA)>
<!ATTLIST client DNI CDATA #IMPLIED> Atribut de l'element "client"
]>
```

Elements intermedis

#PCDATA significa dada de tipus text

1. DTD (Document Type Definition)

1.1. Definició.

Un document XML vàlid que s'ajusta a les restriccions del DTD seria:

```
<clients>
  <client DNI="73042113">
    <IdClient></IdClient>
    <Empresa></Empresa>
    <Ciutat></Ciutat>
    <Provincia></Provincia>
  </client>
</clients>
```

1. DTD (Document Type Definition)

1.1. Elements de Cardinalitat

Parlem de cardinalitat per referir-nos a la quantitat de repeticions què el DTD permet que tingui un element en el document XML.

La cardinalitat s'expressa amb un símbol i en funció d'aquest, s'estableix el nombre de repeticions permeses. A l'exemple anterior podeu veure com l'element "client" té una cardinalitat expressada amb el símbol "+".

1. DTD (Document Type Definition)

1.1. Elements de Cardinalitat

A continuació es mostra la taula de cardinalitats d'un DTD:

SÍMBOL	SIGNIFICAT
+	L'element apareix 1 o més vegades
*	L'element apareix 0 o més vegades
?	L'element apareix 0 o 1 vegada
	L'element apareix 1 vegada a escollir d'una llista

Exemples:

<!ELEMENT Clients (Client *)>	L'element "Clients" conté a zero, un o més "Client"
<!ELEMENT Clients (Client + Referencia Anonim)>	L'element "Clients" pot contenir un o més "Client" o bé una "Referencia" o bé un "Anonim"

1. DTD (Document Type Definition)

1.2. Tipus d'elements i atributs

L'especificació dels continguts d'un **element** pot ser de quatre formes:

- **EMPTY** - Quan és un element buit (no conté elements ni text), encara que pot contenir atributs.

Exemple:

```
<!ELEMENT linia_horitzontal EMPTY>
```

- **ANY** - Permet qualsevol contingut. No es sol utilitzar.
- **MIXED** - Permet contenir tant caràcters com subelements. Exemples:

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

```
<!ELEMENT casa (#PCDATA | habitacle )>
```

- **Element** - Tan sols conté subelements. Exemple:

```
<!ELEMENT ambulatori (recepcio, pediatria, metge_capçalera, infermeria)>
```

1. DTD (Document Type Definition)

1.2. Tipus d'elements i atributs

Tipus d'**atributs**:

- **CDATA** - Pot contenir qualsevol caràcter que respecte les regles de formació.
- **NMTOKEN** - Pot contenir lletres, dígits numèrics, i [., [-], [_], [:]
- **NMTOKENS** - Pot contenir el mateixos caràcters que NMTOKEN a més d'espais en blanc.
- **ID** - Pot contenir els mateixos caràcters que NMTOKEN però ha de ser únic i començar amb una lletra.
- **IDREF** - S'ha de correspondre amb un ID (funciona com les claus alienes en les bases de dades relacionals).
- **IDREFS** - Pot contenir una o més referències a ID's separades per comes.

També podem especificar si els atributs són obligatoris “**#REQUIRED**” o si bé els podem ometre “**#IMPLIED**”.

Podem aplicar valors per defecte als atributs:

<!ELEMENT client (DNI, Nom, Cognom+)>

<!ATTLIST client fumador (si | no) “no”>

1. DTD (Document Type Definition)

1.3. Referència interna i externa a un DTD

Podem efectuar la validació d'un document xml mitjançant un DTD que pot ser referenciat de manera interna. La referència interna vol dir que el DTD es defineix al mateix document xml.

També podem fer referència a un document apart on es troba l'estructura. Aquesta solució té l'avantatge de que es pot re-utilitzar el DTD per a més documents xml simplement fent la referència.

1. DTD (Document Type Definition)

1.3. Referència interna i externa a un DTD

Exemple de referència interna:

```
<? xml version="1.0" ?>
<!DOCTYPE agenda [
  <!ELEMENT agenda (entrada*)>
  <!ELEMENT entrada (nom, cognoms, telefon, adreça)>
  <!ELEMENT nom (#PCDATA)>
  <!ELEMENT cognoms (#PCDATA)>
  <!ELEMENT telefon (#PCDATA)>
  <!ELEMENT adreça (#PCDATA)>
]>
<agenda>
  <entrada>
    <nom></nom>
    <cognoms></cognoms>
    <telefon></telefon>
    <adreça></adreça>
  </entrada>
</agenda>
```

1. DTD (Document Type Definition)

1.3. Referència interna i externa a un DTD

Exemple de referencia externa :

<u>agendaCom.dtd</u>	<u>agendaDeUsuaris.xml</u>
<pre><!ELEMENT agenda (entrada*)> <!ELEMENT entrada (nom, cognoms, telefon, adreça)> <!ELEMENT nom (#PCDATA)> <!ELEMENT cognoms (#PCDATA)> <!ELEMENT telefon (#PCDATA)> <!ELEMENT adreça (#PCDATA)></pre>	<pre><? xml version="1.0" ?> <!DOCTYPE agenda SYSTEM "agendaCom.dtd"> <agenda> <entrada> <nom></nom> <cognoms></cognoms> <telefon></telefon> <adreça></adreça> </entrada> </agenda></pre>

1. DTD (Document Type Definition)

1.3. Referència interna i externa a un DTD


Exercici resolt: Realitza un DTD que validi el següent document XML.

```
<cases>
  <casa portal="1-A">
    <cuina></cuina>
    <aseol></aseol>
    <aseo2></aseo2>
    <salo>124</salo>
    <dormitori1></dormitori1>
    <dormitori2></dormitori2>
  </casa>
  <casa portal="1-B">
    <cuina></cuina>
    <aseol></aseol>
    <aseo2></aseo2>
    <salo></salo>
    <dormitori1></dormitori1>
    <dormitori2></dormitori2>
  </casa>
</cases>
```

1. DTD (Document Type Definition)

1.3. Referència interna i externa a un DTD

Solució:



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE cases {
3   <!ELEMENT cases (casa*)>
4   <!ELEMENT casa (cuina, aseo1, aseo2, salo, dormitori1, dormitori2)>
5   <!ATTLIST casa portal NMTOKEN #REQUIRED>
6   <!ELEMENT cuina (#PCDATA)>
7   <!ELEMENT aseo1 (#PCDATA)>
8   <!ELEMENT aseo2 (#PCDATA)>
9   <!ELEMENT salo (#PCDATA)>
10  <!ELEMENT dormitori1 (#PCDATA)>
11  <!ELEMENT dormitori2 (#PCDATA)>
12 }>
13 <cases>
14   <casa portal="1-A">
15     <cuina></cuina>
16     <aseo1></aseo1>
17     <aseo2></aseo2>
18     <salo>124</salo>
19     <dormitori1></dormitori1>
20     <dormitori2></dormitori2>
21   </casa>
22   <casa portal="1-B">
23     <cuina></cuina>
24     <aseo1></aseo1>
25     <aseo2></aseo2>
26     <salo></salo>
27     <dormitori1></dormitori1>
28     <dormitori2></dormitori2>
29   </casa>
30 </cases>
```

Informació

exercici_1_UF1_NF3.xml és vàlid

1. DTD (Document Type Definition)

1.4. Entitats

Les entitats treballen de forma similar a les constants, es a dir, es defineixen al DTD i després el nom de la entitat queda substituït pel valor que conté.

La declaració al DTD és:

```
<!ENTITY nom_entitat "Eloy">
```

i després utilitzem aquesta entitat amb el format **&nom_entitat;**

```
<nom>&nom_entitat;</nom>
```

1. DTD (Document Type Definition)

1.4. Entitats

Exemple:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE clients[
  <!ELEMENT clients (client+)>
  <!ELEMENT client (IdClient, Empresa, Ciutat, Provincia)>
  <!ELEMENT IdClient (#PCDATA)>
  <!ELEMENT Empresa (#PCDATA)>
  <!ELEMENT Ciutat (#PCDATA)>
  <!ELEMENT Provincia (#PCDATA)>
  <!ATTLIST client DNI CDATA #IMPLIED>
  <!ENTITY autor "Eloy">
]>
<clients>
  <client DNI="73042113">
    <IdClient>&autor;</IdClient>
    <Empresa></Empresa>
    <Ciutat></Ciutat>
    <Provincia></Provincia>
  </client>
</clients>
```

```
-<clients>
  -<client DNI="73042113">
    <IdClient>Eloy</IdClient>
    <Empresa/>
    <Ciutat/>
    <Provincia/>
  </client>
</clients>
```

Observa com l'entitat "autor" es substitueix pel seu valor ...

1. DTD (Document Type Definition)

1.4. Entitats

NOTA:

Fer notar que a l'hora de validar documents XML que contenen espais de noms, no podem fer-ho amb DTD ja que no es va tenir en compte en el moment de la seva creació. La solució és la validació amb XML Schema .

2. XSD (XML Schema Definition)

2.1. Definició

Un XSD ens permet fixar l'estructura que han de seguir els documents XML associats. La diferència bàsica entre XSD i DTD és que el primer utilitza el llenguatge xml per definir la estructura i el DTD no.

Alguns dels avantatges del XSD són que suporta la definició de diferents tipus de dades, com ara, sencers, cadenes, data i hora, etc. També ens permet definir límits màxims i mínims per a les dades i permet l'ús d'espais de noms. No obstant això, també té els seus detractors els quals al·leguen major dificultat d'aprenentatge.

2. XSD (XML Schema Definition)

2.2. Elements simples

Aquests elements no poden contenir atributs ni altres elements al seu interior, sols contenen text (type = integer, string, decimal, boolean, date o time). Es poden definir restriccions per a les dades que contenen.

La seva sintaxis és la següent:

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

```
<xs:element name="dataNaixement" type="xs:date"/>
```

Podem afegir valors "per defecte" a l'element:

```
<xs:element name="color" type="xs:string" default="blau"/>
```

O bé fixar el seu valor per tal que no es pugui modificar:

```
<xs:element name="color" type="xs:string" fixed="vermell"/>
```

2. XSD (XML Schema Definition)

2.2. Elements complexos

Aquests elements poden contenir altres elements i/o atributs.

Exemple:

fitxer.xml
<pre><empleat> <nom>Joan</nom> <cognoms>Valls Lluch</cognoms> </empleat></pre>

fitxer.xsd
<pre><xs:element name="empleat"> <xs:complexType> <xs:sequence> <xs:element name="nom" type="xs:string"/> <xs:element name="cognoms" type="xs:string"/> </xs:sequence> </xs:complexType> </xs:element></pre>

2. XSD (XML Schema Definition)

2.2. Elements complexos

Si ens fixem en la definició de l'esquema, el fet de descriure d'aquesta manera l'element "empleat" implica que només aquest element pot fer ús dels subelements "nom" i "cognoms". A més a més, degut a la seqüència, els subelements apareixeran en l'ordre descrit. Si volem que l'ordre sigui indiferent hem d'utilitzar **<xs:all>** en lloc de **<xs:sequence>**. Si volem limitar el nombre d'ocurrències d'un element podem afegir l'atribut **maxOccurs**, d'altra banda, si volem que es compleixi un mínim, utilitzarem **minOccurs**.

```
<xs:element name="cognom" type="xs:string" minOccurs="1" maxOccurs="2"/>
```

2. XSD (XML Schema Definition)

2.2. Elements complexos

Per tal de re-utilitzar l'estructura que hem assignat a "empleat", hem de fer un petit canvi:

```
<xs:element name="empleat" type="tipus_persona"/>
<xs:complexType name="tipus_persona">
  <xs:sequence>
    <xs:element name="nom" type="xs:string"/>
    <xs:element name="cognoms" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

2. XSD (XML Schema Definition)

2.2. Elements complexos

De manera que podem fer:

```
<xs:element name="empleat" type="tipus_persona"/>
<xs:element name="estudiant" type="tipus_persona"/>
<xs:complexType name="tipus_persona">
  <xs:sequence>
    <xs:element name="nom" type="xs:string"/>
    <xs:element name="cognoms" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

2. XSD (XML Schema Definition)

2.2. Elements complexos

També podem re-utilitzar un tipus complex per crear un altre tipus complex:

```
<xs:element name="empleat" type="persona_completa"/>
<xs:complexType name="tipus_persona">
  <xs:sequence>
    <xs:element name="nom" type="xs:string"/>
    <xs:element name="cognoms" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="persona_completa">
  <xs:complexContent>
    <xs:extension base="tipus_persona">
      <xs:sequence>
        <xs:element name="nom" type="xs:string"/>
        <xs:element name="cognoms" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

2. XSD (XML Schema Definition)

2.3. Atributs

Com s'ha comentat abans, els elements simples no poden contenir atributs, en el moment que afegim un atribut a un element, aquest es considera complex. Per definir un atribut, ho fem de la següent forma:

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

On type = integer, string, decimal, boolean, date o time.

També es pot definir un valor per defecte (default="XXX") o fixe (fixed="XXX") com hem vist als elements i també si és obligatori o no:

```
<xs:attribute name="NIF" type="xs:string" default="12345678A" use="required"/>
```

```
<xs:attribute name="NIF" type="xs:string" use="optional"/>
```

Teniu en compte que l'atribut s'ha de declarar just abans de tancar l'etiqueta </xs:complexType> .

Exemple:

```
<xs:element name="persona">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nom" type="xs:string" />
      <xs:element name="cognoms" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="identificacio" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
```


2. XSD (XML Schema Definition)

2.4. Restriccions

Les restriccions són utilitzades per definir quins valors podem assignar a uns elements determinats i quins no. Anem a veure uns exemples per que quedi més clar:

Exemple 1:

```
<xs:element name="edat">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

El que s'ha definit a l'exemple de dalt és un element simple de tipus sencer que tans sols permet un rang de valors definit entre 0 i 120 ambdós inclosos. Si no volem que s'incloguin, podem canviar "minInclusive" per "minExclusive" i "maxInclusive" per "maxExclusive". La restricció es basa en els elements sencers.

2. XSD (XML Schema Definition)

2.4. Restriccions

Exemple 2: definir el conjunt de valors acceptables per a un element:

```
<xs:element name="cotxe">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="BMW"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Els únics valors acceptables per a l'element "cotxe" són "Audi", "Golf" i "BMW".

2. XSD (XML Schema Definition)

2.4. Restriccions

Exemple 3: indicar unes restriccions a un conjunt de valors ...

```
<xs:element name="inicials">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z][A-Z][A-Z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Aquesta definició també es pot reescriure de la següent manera:

```
<xs:element name="inicials">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z]{3}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

En aquests 2 casos, l'element "inicials" consta de tres caràcters, que poden ser de la "A" majúscula a la "Z" majúscula, per tant no admet altres caràcters, ni tampoc minúscules.

Amb `<xs:pattern value="[a-z]*/>` acceptaríem zero o més lletres minúscules de la "a" fins la "z".

Amb `<xs:pattern value="[a-zA-Z0-9]{8}"/>` acceptaríem 8 caràcters i cadascun d'ells podria ser una lletra de la "a" fins la "z" tant en minúscules com en majúscules o un número entre el 0 i el 9.

Amb `<xs:pattern value="home|dona"/>` acceptaríem només com a valors "home" o "dona".

2. XSD (XML Schema Definition)

2.4. Restriccions

Exemple 4: restriccions en la longitud:

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Hem fet que el password ha de contenir exactament 8 caràcters.

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="5"/>
      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Els password ha de contenir com a mínim 5 caràcters i com a màxim 8.

2. XSD (XML Schema Definition)

2.4. Restriccions

Restriccions:

Restrictions for Datatypes

Constraint	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled

2. XSD (XML Schema Definition)

2.5. Validar un XML amb el seu corresponent Schema

Per tal de validar un document XML amb un esquema que està al mateix directori, utilitzem les capçaleres:

Al document XML :

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

Al document XSD (Schema) **f_vivenda.xsd** :

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

2. XSD (XML Schema Definition)

2.6. Exemple de XML Schema

Creem el document XML anomenat shiporder.xml:

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

<shiporder orderid="889923"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>
  </item>
</shiporder>
```



2. XSD (XML Schema Definition)

2.6. Exemple de XML Schema

Creem l'schema anomenat shiporder.xsd:

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

  <xs:element name="shiporder">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="orderperson" type="xs:string"/>
        <xs:element name="shipto">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="name" type="xs:string"/>
              <xs:element name="address" type="xs:string"/>
              <xs:element name="city" type="xs:string"/>
              <xs:element name="country" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="item" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="title" type="xs:string"/>
              <xs:element name="note" type="xs:string"
minOccurs="0"/>
              <xs:element name="quantity" type="xs:positiveInteger"/>
              <xs:element name="price" type="xs:decimal"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="orderid" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```


2. XSD (XML Schema Definition)

2.6. Exemple de XML Schema

Encara que hauriem pogut afegir:

```
<xs:simpleType name="orderidtype">  
  <xs:restriction base="xs:string">  
    <xs:pattern value="[0-9]{6}"/>  
  </xs:restriction>  
</xs:simpleType>
```