





# Llenguatge de marques i sistemes de gestió de la informació

R.A 4: Estableix mecanismes de validació de documents per a l'intercanvi d'informació utilitzant mètodes per definir-ne la sintaxi i l'estructura.



# DTD

El DTD (*document type definitions*) és un llenguatge de definició d'esquemes que ja existia abans de l'aparició d'XML (es feia servir en SGML). La DTD no segueix la manera de definir els documents d'XML i, per tant, per fer-lo servir, **cal aprendre un nou llenguatge**.

L'objectiu principal de les DTD és proveir un mecanisme per validar les estructures dels documents XML i determinar si el document és **vàlid** o no. Però aquest no serà l'únic avantatge que ens aportaran les DTD, sinó que també els podrem fer servir per compartir informació entre organitzacions, ja que si algú altre té la nostra DTD ens pot enviar informació en el nostre format i amb el programa que hem fet la podrem processar.

El més habitual és afegir la referència a la DTD dins del document XML per mitjà de l'etiqueta especial **<!DOCTYPE**. Com que la declaració XML és opcional però si n'hi ha ha de ser la primera cosa que aparegui en un document XML, l'etiqueta **DOCTYPE** haurà d'anar sempre darrere seu. Es pot declarar tant el principi de tot com després de declarar el document xml:

```
<!DOCTYPE ... >
<?xml version="1.0"?>
```

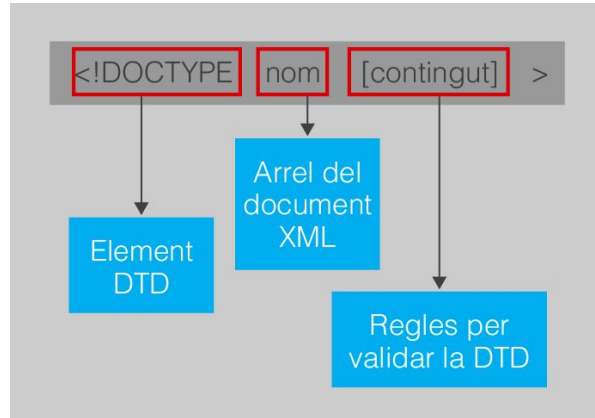
```
<?xml version="1.0" ?>
<!DOCTYPE ... >
<document>
</document>
```

Hi ha dues maneres d'incorporar DTD en un document XML:

- Declaració interna
- Declaració externa

# DTD - Declaració interna

En les declaracions DTD internes les regles de la DTD estan incorporades en el document XML. L'etiqueta **DOCTYPE** es declara com es veu en la figura.



Exemple:

```
<?xml version="1.0"?>
<!DOCTYPE classe [
  <!ELEMENT classe (professor, alumnes)>
  <!ELEMENT professor (#PCDATA)>
  <!ELEMENT alumnes (nom*)>
  <!ELEMENT nom (#PCDATA)>
]>
<classe>
  <professor>Marcel Puig</professor>
  <alumnes>
    <nom>Frededic Pi</nom>
  </alumnes>
</classe>
```

Les declaracions de DTD internes no es fan servir gaire perquè tenen una sèrie de problemes que no les fan ideals:

- En tenir la definició de l'esquema dins del document XML no és fàcil compartir les declaracions amb altres documents.
- Si es tenen molts documents XML, per canviar lleugerament la declaració s'hi hauran de fer canvis en tots.

És per aquest motiu que és més recomanable tenir la DTD en un document a part i fer servir aquest document DTD per validar tots els XML.

# DTD - Declaració externa

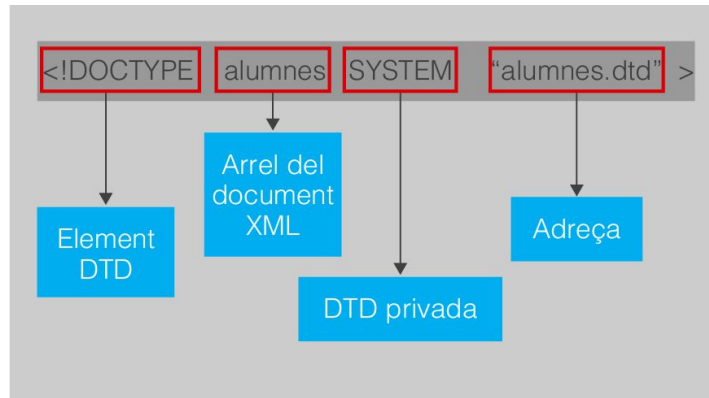
En comptes d'especificar la DTD en el mateix document es considera una pràctica molt millor definir-la en un fitxer a part.

Per fer una declaració externa també es fa servir l'etiqueta `DOCTYPE` però el format és lleugerament diferent i preveu dues possibilitats:

- DTD privada
- DTD pública

# DTD - Declaració externa - Privades

Les definicions de DTD privades són les més corrents, ja que, tot i que es defineixi el vocabulari com a privat, no hi ha res que impedeixi que el fitxer es comparteixi o es publiqui per mitjà d'Internet. La definició d'una DTD privada es fa de la manera que es pot veure en la figura



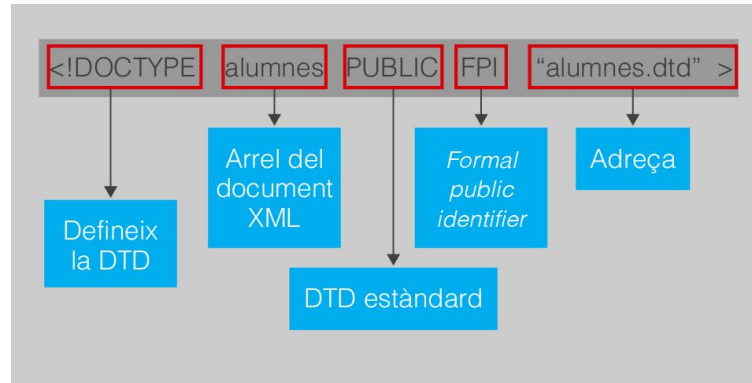
Exemple:

```
<!DOCTYPE alumnes SYSTEM "alumnes.dtd">
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT classe (professor, alumnes) >
<!ELEMENT professor (#PCDATA) >
<!ELEMENT alumnes (nom*) >
<!ELEMENT nom (#PCDATA) >
```

# DTD - Declaració externa - Públiques

Les definicions **PUBLIC** estan reservades per a DTD que estiguin definides per organismes d'estandardització, ja siguin oficials o no. En la definició d'una DTD pública s'afegeix un camp extra que fa d'identificador de l'organisme



La majoria dels camps són idèntics que en definir una DTD privada però en aquest cas s'hi ha afegit un camp més, que és un identificador. L'identificador pot estar en qualsevol format però el més corrent és fer servir el format **FPI** (*formal public identifiers*).

# DTD - Declaració externa - Públiques - FPI

L'**FPI** es defineix per mitjà d'un grup de quatre cadenes de caràcters separades pels dobles barres. En cada posició s'especifica:

```
"simbol//Nom del responsable de la DTD//Document descrit//Idioma"
```

Camp	Valors possibles
primer	Si l'estàndard no ha estat aprovat hi haurà un '-', mentre que si ha estat aprovat per un organisme no estàndard tindrem un '+'. I en canvi, si ha estat aprovat per un organisme d'estàndards, hi haurà una referència.
Nom del responsable	Qui és l'organització o la persona responsable de definir l'estàndard.
Document descrit	Conté un identificador únic del document descrit.
Idioma	El codi ISO de l'idioma en què està escrit el document.

Per exemple, aquesta seria una declaració correcta:

```
<!DOCTYPE classe PUBLIC "-//IOC//Classe 1.0//CA" "http://www.ioc.cat/classe.dtd">
```

# DTD - Regles internes

Qualsevol de les definicions externes també pot contenir un apartat opcional que permet especificar-hi un subconjunt de regles internes.

```
<!DOCTYPE nom SYSTEM "fitxer.dtd" [ regles ] >  
<!DOCTYPE classe PUBLIC "-//IOC//Classe 1.0//CA" "fitxer.dtd" [ regles ] >
```

Si no hi ha regles no cal especificar aquest apartat però també es pot deixar en blanc

```
<!DOCTYPE alumnes SYSTEM "alumnes.dtd" [ ]>
```

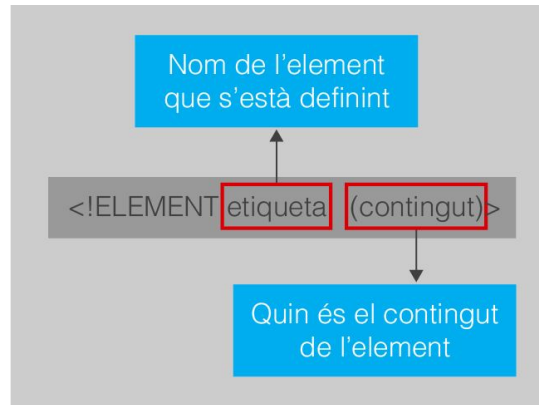
```
<!DOCTYPE alumnes SYSTEM "alumnes.dtd">
```



# DTD - Definició d'esquemes - Elements

La part més important d'un sistema de definició de vocabularis és definir com es fa per determinar l'ordre en què els elements poden aparèixer i quin contingut poden tenir, quins atributs poden tenir les etiquetes, etc. A diferència del que passa en XML, en fer una definició en DTD les etiquetes estan definides. Per definir elements i atributs s'hauran de fer servir etiquetes concretes.

De la mateixa manera que els **elements** són la base de l'XML, la definició d'aquests **elements** és la base de la definició de fitxers DTD. Per tant, sempre s'hauran de definir tots els **elements** que componen el vocabulari i a més especificar-ne el contingut



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

# DTD - Continguts

Com en XML, els continguts poden ser dades o bé altres elements. Podem definir tres grans grups de maneres de definir els continguts en una DTD:

- Continguts genèrics
- Contingut d'elements
- Contingut barrejat

# DTD - Continguts genèrics

Hi ha tres continguts genèrics que es poden fer servir per definir elements: **ANY**, **EMPTY** i **#PCDATA**

Valor	Significat
ANY	El contingut de l'element pot ser qualsevol cosa.
EMPTY	L'element no té contingut.
#PCDATA	El contingut de l'etiqueta poden ser dades.

- **ANY** poden contenir qualsevol cosa dins seu (Tant etiquetes, com dades, o fins i tot una barreja de les dues coses.)

```
<!ELEMENT persona ANY>
```

servirà tant aquest:

```
<persona>Frederic Pi</persona>
```

com aquest:

```
<persona>
  <nom>Frederic</nom>
  <cognom>Pi</cognom>
</persona>
```

- **EMPTY** els elements en XML a vegades no cal que tinguin cap valor per aportar alguna informació, ja que el nom de l'etiqueta pot tenir algun tipus de significat semàntic.

```
<!ELEMENT professor EMPTY>
```

servirà tant aquest:

```
<professor></professor>
```

com aquest:

```
<professor/>
```

- **#PCDATA** (*parser character data*) segurament és el més usat per marcar que una etiqueta només té dades en el seu contingut.

```
<!ELEMENT nom (#PCDATA)>
<!ELEMENT cognom (#PCDATA)>
```

Es pot fer servir **#PCDATA** per definir el contingut dels elements **<nom>** i **<cognom>** perquè dins seu **només tenen dades**. Però no es pot fer servir, en canvi, per definir l'element **<persona>**, perquè dins seu no hi té només dades sinó que hi té els elements **<nom>** i **<cognom>**.

# DTD - Continguts d'elements

Tenim diverses possibilitats per definir el contingut d'elements dins d'una DTD:

- **Seqüències d'elements** es defineix aquesta situació definint explícitament quins són els fills de l'element, separant-los per comes.

```
<!ELEMENT persona (nom,cognom)>
```

Mai no s'ha d'oblidar que les seqüències tenen un ordre explícit i, per tant, només validaran si l'ordre en què es defineixen els elements és idèntic a l'ordre en què apareixeran en el document, XML. (`<cognom>`, `<nom>`) aquest DTD NO validaria

- **Alternatives d'elements** si tenim un XML que defineix president

```
<personal>  
  <president>Josep Maria Flaviol</president>  
</personal>
```

i un XML que

```
<personal>  
  <treballador>Pere Vila</treballador>  
</personal>
```

Podem fer que una DTD validi tots dos casos fent servir l'**operador d'alternativa (|)**. Hi podem incloure més categories i també el valor EMPTY descrit anteriorment.

```
<!ELEMENT personal (treballador|president)>
```

```
<!ELEMENT alumne (delegat|EMPTY)>
```

- **Modificadors** serveixen per especificar quantes instàncies dels elements fills hi pot haver en un element

Modificador	Significat
?	Indica que l'element tant pot ser que hi sigui com no.
+	Es fa servir per indicar que l'element ha de sortir una vegada o més.
*	Indica que pot ser que l'element estigui repetit un nombre indeterminat de vegades, o bé no ser-hi.

```
<!ELEMENT persona (nom,cognom+)>
```

# DTD - Contingut barrejat

El **contingut barrejat** es defineix definint el tipus #PCDATA (sempre en primer lloc) i després s'afegeixen els elements amb l'ajuda de l'operador d'alternativa, |, i s'acaba tot el grup amb el modificador \*.

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

Exemple: Tenim el següent document XML

```
<carta>Benvolgut <empresa>Feros Puig</empresa>:
```

```
Sr. <director>Manel Garcia</director>, li envio aquesta carta per comunicar-li que li hem enviat la seva comanda <comanda>145</comanda> a l'adreça que ens va proporcionar.
```

```
Atentament, <empresa>Ferreteria, SA</empresa>
```

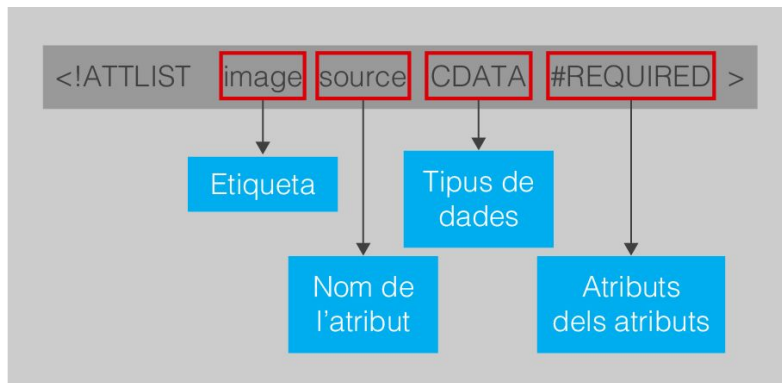
```
</carta>
```

Per tant, podem definir una DTD que validi l'exemple presentat al principi d'aquest apartat de la manera següent:

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

# DTD - Atributs

En les DTD s'han d'especificar quins són els atributs que es faran servir en cada una de les etiquetes explícitament. La declaració d'atributs es fa amb l'etiqueta especial **ATTLIST**, que es defineix tal com es veu en la figura:



Els dos primers valors de la definició de l'atribut són el **nom de l'etiqueta** i el **nom de l'atribut**, ja que en definir un atribut sempre s'especifica a quina etiqueta pertany. Una de les crítiques que s'han fet a les DTD ha estat que no s'hi poden fer atributs genèrics. Si algú vol definir un atribut que sigui compartit per tots els elements del seu vocabulari ha d'anar especificant l'atribut per a tots i cada un dels elements. Per definir un atribut anomenat **nom** que pertanyi a l'element **<persona>** ho podem fer d'aquesta manera:

```
<!ATTLIST persona nom CDATA #IMPLIED>
```

# DTD - Especificar múltiples atributs

Si un element necessita diversos atributs haurem d'especificar tots els atributs en diverses línies **ATTLIST**. Per exemple, definim els atributs **nom** i **cognoms** de l'element **<persona>** d'aquesta manera:

```
<!ATTLIST persona nom CDATA #REQUIRED>  
<!ATTLIST persona cognom CDATA #REQUIRED>
```

O bé es pot fer la definició dels dos atributs amb una sola referència **ATTLIST**:

```
<!ATTLIST persona nom      CDATA #REQUIRED  
                  cognom CDATA #REQUIRED>
```

Les dues ens permetran validar els atributs de l'element **<persona>** d'aquest exemple:

```
<persona nom="Frederic" cognom="Pi" />
```

# DTD - Atribut ATTLIST

Els elements **ATTLIST** a part de tipus de dades també poden tenir atributs que permeten definir característiques sobre els atributs. Aquests atributs es poden veure en la figura:

Atribut	Significat
#IMPLIED	L'atribut és opcional. Els elements el poden tenir o no.
#REQUIRED	L'atribut és obligatori. L'element l'ha de tenir definit o no validarà.
#FIXED	Es fa servir per definir atributs que tenen valors constants i immutables. S'ha d'especificar, ja que és permanent.
#DEFAULT	Permet especificar valors per defecte en els atributs.

```
<!ATTLIST equip posicio ID #REQUIRED>
```

```
<!ATTLIST persona dni NMTOKEN #IMPLIED>
```

```
<!ATTLIST document versio CDATA #FIXED "1.0">  
<!ATTLIST document codificacio NMTOKEN #DEFAULT "UTF-8">
```

s'està obligant que quan es defineixi l'element `<equip>` en un document XML s'especifiqui obligatòriament l'atribut `posicio` i que a més el seu valor no es repeteixi en el document, ja que és de tipus `ID`. En canvi, definint l'atribut `DNI` de `<persona>` amb `#IMPLIED` es permetrà que en crear el document XML l'atribut `DNI` hi pugui ser o no. Els atributs de tipus `#FIXED` han de tenir el valor especificat en la definició i aquest no es pot canviar, mentre que els valors definits amb `#DEFAULT` sí que poden ser canviats.



# DTD - Tipus de dades (I)

Els atributs en DTD no fan servir els tipus de dades dels elements sinó que en defineixen de propis. Els tipus de dades dels atributs es poden veure a la taula

Tipus	Significat
CDATA	Pot contenir qualsevol cadena de caràcters acceptable. Es pot fer servir per a preus, URL, adreces electròniques, etc.
Enumeracions	Es fan servir per definir que l'atribut ha de tenir un dels valors especificats.
ID	L'atribut es podrà fer servir com a identificador d'un element. El seu valor serà únic.
IDREF o IDREFS	El valor són referències a un ID que ha d'existir. El plural és per definir que és una llista.
ENTITY o ENTITIES	Les entitats permeten definir constants per al document i, per tant, el valor de l'atribut ha de ser una entitat.
NMTOKEN o NMTOKENS	Especifiquen cadenes de caràcters que només tinguin caràcters permesos per XML. Per tant, no es permeten els espais.
NOTATION	Permet que l'atribut sigui d'una notació declarada anteriorment.

- **CDATA:** pràcticament idèntic al tipus **#PCDATA** de les etiquetes. En un **CDATA** es pot posar qualsevol dada en format de text tant si té espais com si no en té.

```
<!ATTLIST empresa nom CDATA #REQUIRED>
```

```
<empresa nom="Microsoft Corporation"/>  
<empresa nom="6tem"/>
```

# DTD - Tipus de dades (II)

- **Enumeracions:** Els atributs també poden ser especificats definint-hi quins poden ser els valors correctes per a un atribut. Aquests valors s'especifiquen literalment amb l'operador de selecció:

```
<!ATTLIST mòdul inici (setembre|febrer) #IMPLIED>
```

Per tant, segons la definició, l'atribut **inici** només pot tenir els valors “setembre” o “febrer” i és **implied** (opcional), és a dir, l'atribut pot existir o no en l'element **modul**.

- **ID:** El tipus de dades **ID** serveix per definir atributs que es puguin usar com a identificadors d'un element dins del document. Cal tenir en compte que:
  - Els valors assignats no es poden repetir dins del document. Això els fa ideals per fer servir el tipus **ID** per identificar únicament els elements d'un document.
  - Els valors han de començar per una lletra o un subratllat.

```
<!ATTLIST equip posicio ID #REQUIRED>
```

```
<classificació>  
  <equip posició="primer">F.C.Barcelona</equip>  
  <equip posició="segon">Reial Madrid</equip>  
</classificació>
```

# DTD - Tipus de dades (II)

- **IDREF / IDREFS:** Els valors **IDREF** i **IDREFS** es fan servir per definir valors d'atributs que fan referència a elements que tinguin un valor de tipus **ID**. Només tenen sentit en vocabularis que tinguin atributs amb valor **ID**. Fan referència a l'**ID**.

```
<!ATTLIST recepta id ID #REQUIRED>  
<!ATTLIST ingredient ref IDREF #REQUIRED
```

```
<llibre-cuina>  
  <receptes>  
    <recepta id="recepta1">Patates fregides</recepta>  
    <recepta id="recepta2">Patates bullides</recepta>  
  </receptes>  
  <ingredients>  
    <ingredient ref="recepta1">Oli</ingredient>  
    <ingredient ref="recepta2">Aigua</ingredient>  
  </ingredients>  
</llibre-cuina>
```

IDREFS permet especificar una llista de valors **ID** en el mateix atribut. Per exemple, si es defineix l'atribut *ingredient* amb **IDREFS**:

```
<!ATTLIST ingredient ref IDREFS #REQUIRED>
```

```
<ingredient ref="recepta1 recepta2">Patates</ingredient>
```

# DTD - Tipus de dades (III)

- **NMTOKEN/NMTOKENS:** Els tipus **NMTOKEN** permeten especificar que els atributs poden tenir qualsevol caràcter acceptat per l'XML.

```
<!ATTLIST home naixement NMTOKEN #REQUIRED>
```

```
<home naixement="1970" />
```

El valor en plural **NMTOKENS** permet especificar una llista de valors en comptes d'un de sol:

```
<coordenades posicio="x y"/>
```

- **Notation:** Es fa servir per permetre valors que han estat declarats com a *notation* amb l'etiqueta **<!NOTATION**. Es fa servir per especificar dades no-XML.

```
<!NOTATION GIF SYSTEM "image/gif">  
<!NOTATION JPG SYSTEM "image/jpeg">  
<!NOTATION PNG SYSTEM "image/png">  
<!ATTLIST persona  
  photo_type NOTATION (GIF | JPG | PNG) #IMPLIED>
```

# DTD - Tipus de dades (IV)

- **ENTITY / ENTITIES:** Indica que el valor és una referència a un valor extern que no s'ha de processar. En general solen contenir valors binaris externs.

Fer servir **ENTITY** implicarà haver declarat l'entitat amb **<!ENTITY:**

```
<!ATTLIST persona foto ENTITY #IMPLIED  
<!ENTITY pere SYSTEM "Pere.jpg">
```

Permetrà que es defineixi l'atribut **persona** amb el valor de l'entitat i que automàticament sigui associat a la imatge:

```
<persona nom="pere" />
```

Les etiquetes **<!ELEMENT>** i **<!ATTLIST>** no són les úniques que es poden fer servir per declarar DTD. N'hi ha algunes més que en determinats casos es poden fer servir per crear una DTD. Els més bàsics:

Etiqueta	Es fa servir per ...
<!ENTITY>	Definir referències a fitxers externs que no s'han de processar.
<!NOTATION>	Incloure dades que no siguin XML en el document.
<!INCLUDE>	Fer que parts de la DTD s'afegeixin al processament.
<!IGNORE>	Fer que parts de la DTD siguin ignorades pel processador.

# DTD - Limitacions

El fet que DTD no sigui un llenguatge XML presenta una sèrie de limitacions:

- No comprova el tipus
- Presenta problemes en barrejar etiquetes i `#PCDATA`
- Només accepta expressions deterministes

# DTD - Limitacions - No comprova el tipus

Un dels problemes més importants que ens trobarem a l'hora d'usar DTD per definir el nostre vocabulari és que no té cap manera de comprovar els tipus de dades que contenen els elements. Sovint els noms dels elements ja determinen que el contingut que hi haurà serà d'un tipus determinat (un nombre, una cadena de caràcters, una data, etc.) però la DTD no deixa que se li especifiqui quin tipus de dades s'hi vol posar.

Per tant, si algú emplena amb qualsevol cosa un element que es digui `<dia>`, el document serà vàlid a pesar que el contingut no sigui una data:

```
<dia>xocolata</dia>
```

En no poder comprovar el tipus del contingut, una limitació important afegida és que no hi ha cap manera de poder posar-hi restriccions. Per exemple, no podem definir que volem que una data estigui entre els anys 1900 i 2012.


# DTD - Limitacions - Problemes en barrejar etiquetes i #PCDATA

Una limitació més complexa de veure és que no es poden barrejar etiquetes i **#PCDATA** en expressions si el resultat no és el que es coneix com a “**contingut barrejat**”.

```
<exercici>
  Llegiu el text "Validació de documents XML" i responeu les preguntes següents:
  <apartat numero="1">Què volen dir les sigles XML?</apartat>
  <apartat numero="2">Què és un DTD?</apartat>
</exercici>
```


El més senzill seria mesclar un **#PCDATA** i l'etiqueta **<apartat>**, però és incorrecte:

```
<!ELEMENT exercici (#PCDATA | apartat*)>
```




A més, no es permet que es facin declaracions duplicades d'elements, o sigui que tampoc no ho podem arreglar amb:

```
<!ELEMENT exercici(#PCDATA)>
<!ELEMENT exercici(apartat*)>
```



L'única manera de combinar-ho seria fer servir la fórmula del **contingut barrejat**:

```
<!ELEMENT exercici(#PCDATA|apartat)*>
```





# DTD - Limitacions - Només accepta expressions deterministes (I)

Si ens mirem un document DTD:

```
<!ELEMENT classe(professor|alumnes)>
<!ELEMENT professor (nom,cognoms)>
<!ELEMENT alumnes (alumne*) >
<!ELEMENT alumne (nom,cognom) >
<!ELEMENT nom (#PCDATA)>
<!ELEMENT cognom (#PCDATA)>
```

Quan s'analitza un fitxer XML es defineix quina és l'arrel de la DTD. Si considerem que l'arrel és l'element `<classe>`, el procés de validació començarà en la primera línia que ens diu que perquè el document sigui vàlid després de l'arrel hi ha d'haver un element `<professor>` o bé un element `<alumnes>`. Si el que arriba és un `<professor>` el procediment de validació passarà a avaluar la segona i si arriba un `<alumne>` passarà a la tercera. Si seguim amb l'avaluació veurem que realment el validador sempre que es troba amb una alternativa acabarà anant a una sola expressió. Això és perquè la DTD analitzada és **determinista**.

# DTD - Limitacions - Només accepta expressions deterministes (II)

El mateix ho podem fer amb una expressió més complexa que contingui diferents elements dins de l'alternativa. El validador ha de poder triar, en llegir el primer element, amb quina de les alternatives s'ha de quedar. Si m'arriba un element `<nom>` em quedo amb l'alternativa de l'esquerra, `nom, cognoms`, i si m'arriba un `<àlies>` em quedo amb la de la dreta, `àlies, nom, cognoms`.

```
<!ELEMENT persona(nom,cognom|àlies,nom,cognom)>
```

Si en algun moment algú crea un document que no sigui determinista la validació fallarà. Això passarà si en algun moment el validador es troba que no pot decidir quina és l'alternativa correcta.

```
<!ELEMENT terrestre(persona, home | persona, dona)>
```

Quan des de l'element `<terrestre>` ens arribi un element `<persona>` el processador no tindrà cap manera de determinar si estem fent referència a l'element `<persona>` de l'expressió de la dreta `persona, home` o bé el de l'esquerra `persona, dona`, i per tant fallarà perquè té dues opcions possibles. És una expressió **no determinista**.

Sovint les expressions no deterministes les podem expressar d'una altra manera, de manera que esdevinguin deterministes. L'exemple anterior es podia haver escrit d'una manera determinista perquè en arribar un element `<persona>` no hi hagi dubtes.

```
<!ELEMENT terrestre(persona,(home|dona))>
```

Es força que sempre aparegui un element `<persona>` i després hi haurà l'alternativa entre un `<home>` o un `<dona>` que és determinista.

# DTD - Examples

Anem a veure uns exemples: