UF1 – Programació amb XML

**Nucli Formatiu 1 – A7: Definició d'esquemes i vocabularis a XML: XSD**

XSD significa "Xml Schema Definition"

**Que es un esquema XML?**

* defineix elements que poden aparèixer al document
* defineix atributs que poden aparèixer al document
* defineix que elements son elements fills
* defineix l'ordre dels elements fills
* defineix el numero de elements fills
* defineix si un element es buit o pot incloure text
* defineix tipus de dades pels elements i atributs
* defineix valors per defecte i fixos per elements i atributs

Els esquemes XSD son més potents que altres esquemes com DTD, perquè entre altres coses suporten tipus de dades i fins i tot poden controlar que els seus valors es moguin en rangs predefinits. A més un XSD està escrit segons les regles de XML.

Veiem un **primer exemple** d'un XML i el seu XSD corresponent:

**XML**

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

<email>

<remitent>Roberto</remitent>

<desti>Maria</desti>

<asunto>Llegaré tarde.</asunto>

<texto>El metro esta atascado. Llegare mas tarde de lo previsto.</texto>

</email>

**XSD**

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="0.1" xml:lang="es">

<xs:element name="email">

<xs:complexType>

<xs:sequence>

<xs:element name="remitent" type="xs:string"/>

<xs:element name="desti" type="xs:string"/>

<xs:element name="asunto" type="xs:string"/>

<xs:element name="texto" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

Atenció a com es defineixen els elements simples i els complexos. La declaració de l'element complex és complicada.

1. El element "email" és un tipus complexe perquè conté altres elements.
2. Els elements remitent, destí, asunto i texto són simples perquè no contenen elements fills.

**Com es referencia des del XML al document XSD?**

Dit d'una altra manera, com se li diu que s'ha de validar contra aquell XSD?

S'han d'afegir una serie d'atributs a l'arrel del document XML i aquests atributs són uns o uns altres depenent de si el XSD es troba en una ubicació remota o en local:

Sintaxi per un XSD remot

A l'exemple anterior deixaríem la etiqueta d'obertura <email> així:

<email xmlns="http://www.miweb.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:schemaLocation="http://www.miweb.com email.xsd">

**Sintaxi per un XSD local**

o bé si el xml es troba localment al mateix directori que el XML, així:

**<email xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="email.xsd">**

utilitzarem només aquesta darrera sintaxi a classe.

Què significa cada camp?

xmlns="[http://www.miweb.com](http://www.miweb.com/)"

especifica l'espai de noms per defecte, indicant al validador de XSD que tots els elements que hi han a l'XML estan declarats en aquell espai de noms, és a dir, a l'adreça: [http://www.miweb.com](http://www.miweb.com/).

xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"

especifica l'espai de noms pel qual es valida el document XSD, és a dir, les seves etiquetes XML perquè no oblidem que el XSD també és un XML.

xsi:schemaLocation="http://www.miweb.com email.xsd"

especifica 2 valors separats per un espai:

* el primer es l'espai de noms a utilitzar
* el segon la localització del fitxer XSD que hi ha dins.

xsi:noNamespaceSchemaLocation="email.xsd"

especifica la ruta del fitxer XSD local contra el que s'ha de validar.

**Element <xs:schema>**

És l'arrel d'un document XSD, l'estructura bàsica del qual és la següent:

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
targetNamespace="http://www.miweb.com"  
xmlns="http://www.miweb.com"  
elementFormDefault="unqualified">

...

</xs:schema>

xmlns:xs="http://www.w3.org/2001/XMLSchema"

indica que els elements i tipus de dades utilitzades a l'esquema venen d'aquella URL i també indica que els elements i tipus de dades que venen d'aquella URL han d'escriure amb el prefix "xs:".

targetNamespace="http://www.miweb.com"

indica que els elements definits en aquest esquema (email,remitent,desti, asunto,texto) pertanyen a l'espai de noms "http://www.miweb.com".

xmlns="[http://www.miweb.com](http://www.miweb.com/)"

indica que l'espai de noms per defecte és aquesta URL

* elementFormDefault="qualified"
  + indica que qualsevol element utilitzat pel XML el qual estigui declarat en aquest esquema ha de ser qualificat pel espai de noms.
  + "qualified" significa aquí que tots els elements i atributs o bé estan al targetNamespace definit o bé porten el prefixe declarat.
  + "unqualified" significa que els elements i atributs no es troben a cap espai de noms.

**Elements simples**

Son elements XML que només poden contenir text, entenent text en un sentit ampli ja que aquí, text, pot significar :

* string,
* integer,
* date,
* booleà o
* un tipus de dades definit per nosaltres mateixos.

Es poden afegir restriccions a un tipus de dades per tal de:

1. limitar el seu contingut
2. per obligar a que la data tingui una determinada forma.

La definició d'un element simple és:

<xs:element name="nom" type="tipus\_elem"/>

Els **tipus de dades** més normals son:

* xs:string
* xs:decimal
* xs:integer
* xs:boolean
* xs:date
* xs:time

Els elements simples poden tenir un valor per defecte o un valor fixe(que no es pot canviar). El primer s'assigna a l'element quan no s'especifica un valor en ell.

Sintaxi per definir el valor d'un element per defecte:

<xs:element name="tiempo" type="xs:string" default="nuboso"/>

Un valor fixe s'assigna automàticament a l'element i no pots especificar un altre.

Sintaxi per definir el valor fixat d'un element:

<xs:element name="tiempo" type="xs:string" fixed="nuboso"/>

|  |
| --- |
| **Primer Exercici:**  Fer un XSD per validar el següent XML, tot fent que si no s'indica cap ciutat , el seu valor per defecte sigui "Terrassa":  <?xml version="1.0" encoding="UTF-8"?>  <persona>  <nom>posar aqui el vostre nom</nom>  <edat>posar aqui la vostra edat</edat>  <datanaixement> posar aqui la vostra data de naixement</datanaixement>  <ciutat>posar aqui la vostra ciutat</ciutat>  </persona> |

**Atributs**

Tots els atributs son d'un tipus de dades simple.

La sintaxi per definir un atribut es:

<xs:attribute name="nom\_atribut" type="tipus\_atribut"/>

Els **tipus de dades** per atributs són:

* xs:string
* xs:decimal
* xs:integer
* xs:boolean
* xs:date
* xs:time

Per exemple, el següent atribut «model»:

<avio model="NG300">Airbus</avio>

es definiria així al XSD:

<xs:attribute name="model" type="xs:string"/>

Els atributs poden tenir un valor per defecte o un valor fixe especificat (que no es pot canviar). El valor per defecte s'assigna a l'atribut quan no s'especifica un valor en ell.

Sintaxi per definir el valor d'un atribut per defecte:

<xs:attribute name="model" type="xs:string" default="NG100"/>

Un valor fixe s'assigna automàticament a l'atribut i no pots especificar un altre.

Sintaxi per definir el valor fitxat d'un atribut:

<xs:attribute name="model" type="xs:string" fixed="NG100"/>

També es pot definir si un atribut es opcional o obligatori:

Sintaxi:

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

ó

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

**Però... on es col·loca la definició de l’atribut?**

Depèn de si es tracta d’un element complex o d’un element simple.

A dins d’un element complex:

<xs:element name="persona">

<xs:complexType>

<xs:sequence>

<xs:element name="nom" type="xs:string"/>

</xs:sequence>

**<xs:attribute name="curs" type="xs:string"/>**

</xs:complexType>

</xs:element>

A dins d’un element simple:

<xs:element name="avio">

<xs:complexType>

<xs:simpleContent>

<xs:extension base="xs:string">

**<xs:attribute name="model" type="xs:string"/>**

</xs:extension>

</xs:simpleContent>

</xs:complexType>

</xs:element>

|  |
| --- |
| **Exercici:**  Feu el XML+XSD de la següent informació d'un alumne d'una escola:   * codi d'alumne (número enter) * nom * cognom1 * cognom2 * dni * data de naixement * curs * grup * quota pagada (booleà)   Quan valideu el XML contra el XSD correctament, feu que "dni" sigui un atribut de l'element "alumne". Especifiqueu també que l'atribut "dni" és obligatori. |

**Restriccions**

Quan definim un element senzill o atribut d'un determinat tipus de dades, això imposa una restricció al seu contingut.

Per això, podem definir les nostres restriccions pròpies.

**Restriccions sobre el valor**

Podem restringir el rang de valors vàlids per a un determinat element.

Per exemple suposem que volem restringir que un element "edat" tingui valors entre 0 i 100.

<xs:element name="edat">

<xs:simpleType>

<xs:restriction base="xs:integer">

<xs:minInclusive value="0"/>

<xs:maxInclusive value="100"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

**Restriccions sobre un conjunt de valors**

Podem restringir el rang de valors vàlids per a un determinat element. Diguem que definim un conjunt amb els valors vàlids per aquest element.

Per exemple suposem que volem restringir que un element "subgrupde1ASIX" tingui valors del conjunt {"A","B"}.

<xs:element name="subgrupde1ASIX">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:enumeration value="A"/>

<xs:enumeration value="B"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

**Restriccions sobre una serie de valors**

Podem restringir els valors vàlids per a un determinat element definint una serie de números o lletres que poden ser utilitzats. Dit d'una altra manera definim un patró per aquest element.

Per exemple, suposem que estem guardant dades sobre els pisos d'una escala. Lògicament 2 elements que hi hauran seran:

* planta (per guardar si es un primer, segon, ... o desé)
* lletra (per guardar quina es la lletra de cada pis dins una mateixa planta: A, B, C,D,E,...) .

Volem restringir aquest últim element "lletra" tingui valors del conjunt de lletres majúscules, és a dir des d' "A" fins a "Z"}.

La restricció es definiria així:

<xs:element name="lletra">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="[A-Z]"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

Segons l'anterior definició, l'element "lletra" podrà prendre el valor d'una única lletra majúscula, per exemple : "C" o "T" o "U", etc.

Que passa si hem de guardar més d'una lletra?

Imagineu per exemple que necessiteu guardar una matricula de cotxe que es composa de 4 números i 3 lletres. Si guardem lletres i números per separat , el element que guarda les 3 lletres s'haurà de definir així:

<xs:element name="lletresdematricula">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="[A-Z][A-Z][A-Z]"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

Definicions d'altres possibles patrons:

Restringeix a una lletra minúscula o majúscula el valor de cadascuna de les 3 lletres que hi ha d'haver:

<xs:element name="treslletres">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

Com especifiquem a l’expressió regular que defineix el pattern d'un determinat element que una determinada posició [ ] té un nombre exacte d’ocurrències? O una o més ocurrències?; O cero o més ocurrències?; O que únicament pot haver-hi 0 o 1? ocurrències?. Escrivint immediatament tancar el [ ] un dels següents caràcters:

* '' (res) : l'element apareix obligatòriament una vegada.
* '{n}' : ha d’haver exactament ‘n’ ocurrències de l’element, ni més ni menys.
* '+': pot haver 1 o més ocurrències de l'element.
* '\*': pot haver 0 o més ocurrències de l'element.
* '?': pot haver 0 o 1 ocurrència de l'element.

Per guardar un número de telèfon(que son 9 xifres numèriques) , s'ha de restringir el numero de posicions i el valor de cadascuna d'elles:

<xs:element name="telefon">

<xs:simpleType>

<xs:restriction base="xs:integer">

<xs:pattern value="[0-9]{9}"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

Restricció que no controla la longitud del contingut però si que tot siguin lletres:

<xs:element name="frase">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="([a-z])+"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

Així hem definit que l'element pot tenir 0 o més caràcters però que tots han de ser lletres minúscules.

Però l'anterior no és molt realista donat que no acceptarà una frase normal amb espais en blanc entre les paraules.

El caràcter d'espai en blanc es representa per \s

De manera que si, dins un patró, necessiteu fer servir l'espai en blanc, podeu fer-ho així:

<xs:element name="fraseambespais">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="([a-z\s])\*"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

De vegades necessitem especificar un determinat rang possible de valors però excloure’n un determinat subrang seu; és a dir, imaginem que volem fer un element que guardi només consonants (sense permetre’n cap vocal):

<xs:element name="consonants">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="[a-z-[aeiou]]\*"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

A l’anterior exemple, el rang exterior defineix el conjunt general de caràcters que validem mentre que el rang interior (el que es troba després del menys (-)), és el subconjunt de caràcters que no volem permetre.

Restricció per triar entre 2 valors concrets, per exemple si volem guardar el sexe d'una persona (home o dona).

<xs:element name="sexe">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="home|dona"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

**Restricció per limitar el valor d'un element a un numero concret de caràcters.**

Imaginem per exemple que hem de fer una contrasenya de 10 caràcters, definiríem aquest element així:

<xs:element name="micontrasenya">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="10"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

I ara imaginem que també hi ha una longitud mínima per la contrasenya, que no pot ser menys de 4 caràcters ni més de 10.

<xs:element name="micontrasenya">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:minLength value="5"/>

<xs:maxLength value="10"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

**Taula Resum de Restriccions**

|  |  |
| --- | --- |
| **Restricció** | **Significat** |
| enumeration | Defineix una llista de valors acceptables. |
| fractionDigits | Especifica el numero màxim de decimals permesos a un numero. Ha de ser més gran o igual que 0. |
| length | Especifica el numero de caràcters permès. Ha de ser més gran o igual que 0. |
| maxExclusive | Especifica el límit superior per valors numèrics (el valor ha de ser menor a aquest valor). |
| maxInclusive | Especifica el límit superior per valors numèrics(el valor ha de ser menor a aquest valor). |
| maxLength | Especifica el numero màxim de caràcters permès. Ha de ser més gran o igual que 0. |
| minExclusive | Especifica el límit inferior per valors numèrics(el valor ha de ser més gran a aquest valor). |
| minInclusive | Especifica el límit inferior per valors numèrics(el valor ha de ser més gran o igual a aquest valor). |
| minLength | Especifica el numero mínim de caràcters permès. Ha de ser més gran o igual que 0. |
| pattern | Defineix un patró, o sigui, la seqüència exacta de caràcters acceptables. |
| totalDigits | Especifica el número exacte de dígits permès. Ha de ser més gran o igual que 0. |
| whiteSpace | Especifica com tractar els espais en blanc que hi ha al contingut. S'entén que els tabuladors i els avançaments de línia també son espais en blanc. |

**Elements complexes**

Un element complexe és aquell que co nté algun altre element a dins seu.

Per exemple aquest xml:

<persona>

<nom>alfredo</nom>

<cognom>bercero</cognom>

</persona>

L'element "persona" és complexe i els elements "nom" i "cognom" son elements simples. Es declara en XSD de la següent manera:

<xs:element name="persona">

<xs:complexType>

<xs:sequence>

<xs:element name="nom" type="xs:string"/>

<xs:element name="cognom" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:element>

Atenció: <xs:sequence> obliga a que els elements que es declaren a continuació es trobin al XML en el mateix ordre en que es defineixen aquí.

**Indicadors**

Hi han una serie d'Indicadors que afecten als elements fills d'un element complexe, alguns afecten a l'ordre dels fills i d'altres al numero que hi pot haver.

Els indicadors d'ordre son: "all" , "choice" i "sequence".

**Indicador "all"**

Indica que els elements fills poden aparèixer en qualsevol ordre i que cada element fill ha d'aparèixer una única vegada.

Exemple:

<xs:element name="persona">

<xs:complexType>

<xs:all>

<xs:element name="nom" type="xs:string"/>

<xs:element name="cognom" type="xs:string"/>

</xs:all>

</xs:complexType>

</xs:element>

Segons l'anterior definició aquest XML serà correcte:

<persona>

<cognom>bercero</cognom>

<nom>alfredo</nom>

</persona>

**Indicador "choice"**

Indica que pot aparèixer un element fill o un altre.

Exemple:

<xs:element name="persona">

<xs:complexType>

<xs:choice>

<xs:element name="fixe" type="xs:integer"/>

<xs:element name="mobil" type="xs:integer"/>

</xs:choice>

</xs:complexType>

</xs:element>

Segons l'anterior definició aquest XML serà correcte:

<persona>

<mobil>666666666</mobil>

</persona>

<persona>

<fixe>666666666</fixe>

</persona>

**Indicador "sequence"**

Indica que els elements fills han d'aparèixer en el mateix ordre en que son definits.

Exemple:

<xs:element name="persona">

<xs:complexType>

<xs:sequence>

<xs:element name="nom" type="xs:string"/>

<xs:element name="cognom" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:element>

Segons l'anterior definició aquest XML serà erroni:

<persona>

<cognom>bercero</cognom>

<nom>alfredo</nom>

</persona>

Els indicadors d'ocurrències son: "maxOccurs" i "minOccurs"

**Indicador "maxOccurs"**

Indica que el numero màxim de vegades que un element fill pot aparèixer. Si el numero d'ocurrències ha de ser il·limitat llavors el valor ha de ser "unbounded".

Exemple:

<xs:element name="persona">

<xs:complexType>

<xs:sequence>

<xs:element name="nom" type="xs:string"/>

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

</xs:sequence>

</xs:complexType>

</xs:element>

Segons l'anterior definició aquest XML serà correcte:

<persona>

<nom>alfredo</nom>

<cognom>bercero</cognom>

<cognom>gracia</cognom>

</persona>

**Indicador "minOccurs"**

Indica que el numero mínim de vegades que un element fill pot aparèixer.

Exemple:

<xs:element name="persona">

<xs:complexType>

<xs:sequence>

<xs:element name="nom" type="xs:string"/>

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

<xs:element name="mobil" type="xs:integer" minOccurs="0" maxOccurs="10"/>

</xs:sequence>

</xs:complexType>

</xs:element>

Segons l'anterior definició aquest XML serà correcte:

<persona>

<nom>alfredo</nom>

<cognom>bercero</cognom>

<cognom>gracia</cognom>

<mobil>654423781</mobil>

<mobil>654423781</mobil>

<mobil>654423781</mobil>

<mobil>654423781</mobil>

<mobil>654423781</mobil>

<mobil>654423781</mobil>

<mobil>654423781</mobil>

<mobil>654423781</mobil>

<mobil>654423781</mobil>

<mobil>654423781</mobil>

</persona>

|  |
| --- |
| **Exercici:**  Teniu un concessionari de cotxes i voleu emmagatzemar en un XML les característiques de cadascú. Una vegada fet el XML cal definir el XSD d'acord al que s'ha explicat a classe per de complir les següents especificacions/restriccions:   * La etiqueta arrel ha de ser <cotxes> * La etiqueta per guardar les informacions de cada cotxe s'ha de dir: <cotxe> * Les informacions a guardar de cada cotxe son les següents:   + marca   + modelo   + matricula   + numero de portes   + cilindrada   + dni de cada propietari que el cotxe hagi tingut. * La marca serà sempre francesa, ja sabeu quines son les 3 marques de cotxes franceses més conegudes. * La matrícula ha de complir el format de les matrícules espanyoles noves (4 dígits més 3 lletres). * El numero de portes d'un cotxe no pot ser qualsevol número. Penseu en quantes portes pot tenir un cotxe. * La cilindrada ha de mantenir-se en uns valors raonables. * El concessionari ven cotxes nous i cotxes de segona mà, de forma que en aquest últim cas cada cotxe pot tenir un numero aleatori de propietaris previs. Si n'ha tingut s'ha de guardar el dni de cadascun d'ells. Com ja sabeu el dni segueix un format estricte que haureu de controlar que és correcte. |

**Tipus personalitzats de dades**

Com hem vist, les restriccions ens permeten personalitzar el tipus de dades concret que ha de tenir un elements, ja que de vegades necessitem representar tipus de dades que no es corresponen amb els predefinits (integer, string, boolean, etc.).

De vegades necessitem aplicar la mateixa personalització a diferents elements i atributs dins un mateix XML, i això provoca repeticions de codi innecessàries.

Observeu el següent XML d’una associació de mares i pares d’alumnes d’una escola:

<AMPA percentatgePressupostPublic="20">

<familiesSocies percentatge="88"/>

<menjador assistencia="43"/>

<extraescolars participacio="31"/>

</AMPA>

Com veieu, a tots els atributs els hem d’aplicar exactament la mateixa restricció per garantir que el percentatge sigui vàlid:

<xs:attribute name="percentatgePressupostPublic">

<xs:simpleType>

<xs:restriction base="xs:integer">

<xs:minInclusive value="0"/>

<xs:maxInclusive value="100"/>

</xs:restriction>

</xs:simpleType>

</xs:attribute>

<xs:attribute name="percentatge">

<xs:simpleType>

<xs:restriction base="xs:integer">

<xs:minInclusive value="0"/>

<xs:maxInclusive value="100"/>

</xs:restriction>

</xs:simpleType>

</xs:attribute>

<xs:attribute name="assistencia">

<xs:simpleType>

<xs:restriction base="xs:integer">

<xs:minInclusive value="0"/>

<xs:maxInclusive value="100"/>

</xs:restriction>

</xs:simpleType>

</xs:attribute>

<xs:attribute name="participacio">

<xs:simpleType>

<xs:restriction base="xs:integer">

<xs:minInclusive value="0"/>

<xs:maxInclusive value="100"/>

</xs:restriction>

</xs:simpleType>

</xs:attribute>

Per tal d’evitar l’anterior repetició de codi, podem extreure el tipus de dades a l’exterior, declarant-lo per separat i després cridar-lo pel seu nom; aquí hi ha com el faríem:

*<!-- Declaració del tipus de dades -->*

<xs:simpleType name="**tantXcent**">

<xs:restriction base="xs:integer">

<xs:minInclusive value="0"/>

<xs:maxInclusive value="100"/>

</xs:restriction>

</xs:simpleType>

*<!-- Utilització -->*

<xs:attribute name="percentatgePressupostPublic" type="**tantXcent**"/>

<xs:attribute name="percentatge" type="**tantXcent**"/>

<xs:attribute name="assistencia" type="**tantXcent**"/>

<xs:attribute name="participacio" type="**tantXcent**"/>

A l’interior de <xs:simpleType>, a més d’utilitzar <xs:restriction> per definir restriccions sobre els tipus de dades bàsics, tenim més opcions com ara: list i union.

Utilitzar <xs:simpleType> en combinació amb <xs:list> ens permet definir un element que ha de contenir una llista de valors:

*<!-- Declaració del tipus de dades -->*

<xs:simpleType name="**nombresDecimals**">

<xs:list itemType="xs:decimal"/>

</xs:simpleType>

*<!-- Utilització del tipus de dades -->*

<xs:element name="mesuraAltAmpleFons" type="**nombresDecimals**"/>

*<!-- XML -->*

<mesuraAltAmpleFons>6.34 2.11 3.35</ mesuraAltAmpleFons>

Utilitzar <xs:simpleType> en combinació amb <xs:union> ens permet definir un element d’un tipus de dades que aplega diferents tipus anteriorment definits:

<!-- Declaració del tipus de dades -->

<xs:simpleType name="**udLongitudImperial**">

<xs:restriction base="xs:string">

<xs:pattern value="legua|milla|yarda|pie|pulgada"/>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="**udLongitudSMD**">

<xs:restriction base="xs:string">

<xs:pattern value="km|hm|dam|m|dm|cm|mm"/>

</xs:restriction>

</xs:simpleType>

<!-- union de 2 tipus de dades -->

<xs:simpleType name="**udLongitud**">

<xs:union memberTypes="**udLongitudImperial** **udLongitudSMD**"/>

</xs:simpleType>

*<!-- Utilització del tipus de dades -->*

<xs:element name="longitud" type="xs:positiveInteger"/>

<xs:element name="unitat" type="**udLongitud**"/>

*<!-- XML --> ó <!-- XML -->*

<objecte> <objecte>

<longitud>34</longitud> <longitud>45</longitud>

<unitat>pie</unitat> <unitat>cm</unitat>

</objecte> </objecte>

|  |
| --- |
| **Exercici d'aplicació**  Feu el XML+XSD de la següent informació d'un alumne d'una escola:   * codi d'alumne **(número enter positiu)** * nom * cognom1 * cognom2 * dni **(ben format)** * data de naixement **(amb el nostre format DD/MM/AAAA)** * curs **(enter vàlid pels estudis d'ASIX)** * quota pagada (booleà) * **edat (acotada lògicament)** * **telèfon (format correcte)** * **tallaPantalons (talla real)** * **tallaCamisa (talla real)** * **tallaSamarretaTècnica (talla real)** * **tallaJaqueta (talla real)**   Aquí teniu una taula resum dels diferents sistemes de talles real de món: |

Referència Software d'edició/validació:

* XMLcopy
* XMLNotepad,
* TotalEdit,
* SernaFree,
* xmlPro (Vervet Logic)
* <oXygen/>

Referència Validadors online de XML:

* The W3C Markup Validation Service
* XML Validation
* Validome