Tema 3. Validació de documents XML (XSD)

Índex de continguts

3.1. Esquemes a XML, una introducció (XSD i DTD).

Una vegada el nostre document XML és correcte sintàcticament, veurem com validar-lo. Els esquemes serveixen per a comprobar que un document XML és vàlid respecte a un conjunt de regles ben definit. Donat el següent XML:

```
<llibre isbn="0-2777-6861-6">
  <autor>J.D. Salinger</autor>
  <titol>The Catcher in the Rye</titol>
</llibre>
```

Un esquema permet validar:

- L'**estructura dels elements i els atributs.** Per exemple, l'element **llibre** ha de contenir els elements **autor** i **titol**, i opcionalment l'atribut **isbn**.
- L'**ordre dels elements**. Per exemple, podem especificar que l'element **autor** va abans que **titol**,
- **Tipus de dades** dels elements i els atributs. Enter, decimal o cadena de texte són alguns exemples. També es poden definir tipus de dades basades

en rangs, enumeracions o coincidència de patrons (patterns). Per exemple, podem definir que l'atribut **isbn** tingui el patró 0-0000-0000-0.

Els esquemes XML poden tenir altres utilitats:

- **Servir com a contracte entre empreses que es suministren productes**. Un esquema indica clarament quins elements han d'aparèixer en els documents que rebrà. Per exemple, pot especificar que totes les factures indiquin el NIF de l'empresa a qui es factura.
- **Documentació de sistema**. Permet a qualsevol que vulgui, entendre els noms, atributs i estructura del document XML. També es poden fer anotacions.
- **Incorporació de dades noves** XSD permet incorporar valors per defecte i fixes al document XML. També permet normalitzar els espais en blanc.

XML Schema o XSD (XML Schema Definition) permet definir de forma molt precisa el contingut dels documents XML. Existeixen altres llenguatges d'esquema:

- Relax NG. Basat en gramàtica. És més fàcil d'entendre que XML Schema. Desenvolupat per Oasis.
- Schematron. Està basat en afirmacións enlloc de gramàtica.

També hi ha utilitats per convertir d'aquests formats a XSD.

3.1.1. DTD i XSD

Un document ben format pot ser validat amb les tecnologies DTD (Document Type Definition) o XSD (XML Schema).

DTD defineix els blocs o elements d'un document XML. DTD prové d'un subconjunt del llenguatge SGML. Des de principis de 2000, DTD ha estat substituit per altres opcions com XSD o Relax NG, principalment degut a les seves limitacions.

Algunes de les seves limitacions són:

- Un document DTD no és un document XML, per tant no podem verificar que estigui ben format.
- No es poden fer restriccions sobre els diferents valors que pot pendre un element: tamany, tipus de dades, etc.
- No es pot donar un valor per defecte per als elements (si els atributs).
- No soporta espais de noms.

L'alternativa als documents DTD són els esquemes XML o XSD (XML Schema Definition).

XSD supera les limitacions de DTD:

- Els documents XSD deriven de XML, per tant són documents XML i es poden comprobar sintàcticament.

- XSD defineix molts tipus de dades predefinits.
- XSD permet definir la cardinalitat dels elements.
- XSD permet mesclar diferents vocabularis XML (espais de noms).

Els documents XSD tenen l'inconvenient de que són lleugerament més difícils d'interpretar que els DTD.

Un Esquema XML és el motlle d'on sortiran els diferents documents XML que compliran l'estructura definida a l'esquema, cadascun amb les seves dades concretes.

Per validar els documents XML, podem fer servir les eines de programació vistes fins ara. També podem trobar alguns validadors on-line, com per exemple:

CoreFiling XML Schema Validator

3.2. Estructura d'un esquema XML.

Per crear un esquema haurem de crear un document amb extensió XSD que validarà el document XML. Per indicar-li a un document XML quin document de validació volem fer servir, afegirem un atribut a l'element arrel del document XML.

3.2.1. Regles XSD

Un esquema XML és un document XML que ha de complir les següents regles:

- L'element arrel s'anomena **schema**.
- L'espai de noms ha de ser http://www.w3.org.2001/XMLSchema. Es pot no usar prefixe, utilitzar xs o xsd:

```
<?xml versión="1.0"?>
<xs:schema xmnls:xs ="http://www.w3.org.2001/XMLSchema">
.....
</xs:schema>
```

- Igual que a DTD, el primer que hem de fer és declarar l'element arrel del nostre XML. La sintaxi de **xs:element** seria:

```
<xs:element name = "nom-element-arrel">
```

  Aquest element conté l'atribut name que defineix l'element arrel del document. XSD permet tenir més d'un element arrel al nostre esquema.

```
<?xml versión="1.0"?>
  <xs:schema xmlns:xs ="http://www.w3.org.2001/XMLSchema">
  <xs:element name="cotxe" />
  <xs:element name="vaixell" />
</xs:schema>
```

- L'ordre en que es declaren els elements (anomenats **components** a XSD) no és rellevant.

3.2.2. Vinculació d'un document XML amb un document XSD.

- Per a vincular un esquema amb un document XML hem d'afegir l'atribut **noNamespaceSchemaLocation** a l'element arrel **del nostre document XML**:

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

En l'exemple anterior, a l'arxiu **landrover.xsd** tindrem les nostres declaracions XSD. Els blocs principals d'un esquema XSD són **xs:element** i **xs:attribute**, que veurem a continuació.

3.2.3. Elements

Els elements i els atributs són els blocs principals amb els que es construeixen els documents XML. Per tant, el nostre esquema XSD contindrà una declaració per a cada element i cada atribut que hi hagi al document.

Cadascun dels elements i atributs està associat a un **tipus de dades**. XSD separa els conceptes d'element i atributs dels seus tipus de dades. Això permet reutilitzar la mateixa estructura a varis elements idèntics. Per exemple, a un a factura podem tenir dos elements anomenats **direccionEnvio** i **direccionFacturacion** que tenen la mateixa estructura pero noms diferents. Només hem de declarar un tipus **TipoDireccion** i fer-lo servir en la declaració dels elements.

Els elements han d'incloure com a mínim l'atribut name per especificar el nom de l'element XML i type per especificar el tipus de dades.

Exemples de declaració d'elements a XSD:

```
<xsd:element name="tamany" type="TamanyTipus"/>

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

<xsd:element name="product">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="name"/>
      <xsd:element ref="size"/>
    <xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="anything"/>
```

El primer element fa servir el tipus **TamanyTipus** per indicar el tipus de dades de **tamany**.

També podem definir un tipus de dada especificant **simpleType** o **complexType**.

Per últim, si no hem especificat cap tipus de dades, el tipus serà **anyType** que permet qualsevol contingut amb fills o caràcters.

Els atributs principals de **<xs:element>**:

nom atribut	propòsit
name	nom de l'element, obligatori si és l'arrel
ref	nom d'un altre element que es troba en el document
type	el tipus d'element (veure tipus de dades)
default	valor per defecte de l'element. Només es pot utilitzar per a continguts amb texte.
fixed	indica l'únic valor possible, sempre que sigui texte.
minOccurs	número mínim d'ocurrències d'un element. veure indicadors.
maxOccurs	número màxim d'ocurrències d'un element. veure indicadors.
id	identificador únic per a l'element

3.2.4. Tipus de dades

Els tipus de dades més comuns a XSD són:

- xs:string (cadena)
- xs:decimal (número decimal)
- xs:integer (sencer)
- xs:boolean (veritable/fals)
- xs:date (data)
- anyURI (adreça web)

Exemples:

```
<xs:element name="dia" type="xs:date" /> <dia>2011-09-15</dia>
<xs:element name="altura" type="xs:integer"/> <altura>220</ altura >
<xs:element name="nombre" type="xs:string"/> < nombre >Pere Puig</nombre >
```

```
<xs:element name="tamaño" type="xs:float"/> <tamaño>1.7E2</tamaño>
```

Activitat 8. Explica el següent codi:

```
<?xml version="1.0"?>
  <xs:schema xmnls:xs="http://www.w3.org.2001/XMLSchema">
  <xs:element name="coche" type="xs:string" maxOccurs="3"/>
  <xs:element name="nombre" type="xs:string"/>
  <xs:element name="edad" type="xs:integer" default="18"/>
  <xs:element name="permisoConducir" type="xs:boolean" default="true"/>
</xs:schema>
```

3.2.5. Atributs

Fem servir el component **<xs:attribute>** per especificar els atributs dels nostre document XML.

Atributs principals de **<xs:attribute>**:

nom atribut	propòsit
name	nom de l'atribut. Aquest atribut no puede aparèixer al mateix temps que ref
ref	referència a la descripció de l'atribut que es troba en altre lloc de l'esquema. Si surt aquest atribut, no apareixeran els atributs type, ni form, ni podrà contenir un component xs:simpleType
type	el tipus d'element (veure tipus de dades)
use	indica si l'existència de l'atribut és opcional, obligatòria o prohibida (optional, required, prohibited)
default	valor que prendrà l'atribut quan sigui procesat per alguna aplicació quan al document XML no hi té cap valor assignat. No pot aparèixer al mateix temps que fixed
fixed	Valor únic que pot contenir el document. No pot aparèixer simultàniament amb default
id	indica identificador únic per al atribut

Exemple:

```
<xs:attribute name="alias" type="xs:string"/>
<!-- Ahora podemos usarlo dentro de un elemento nombre -->
<nombre alias="Snake"> Plissken </nombre>
```

Un exemple complet:

XML Schema Tutorial

Activitat 9.

Crea l'atribut moneda de tipus cadena que per defecte tingui el valor euro. Un atribut unitat amb tipus de dades cadena que tingui un valor fixe "minuts" i un element idEmple amb tipus de dades enter positiu i amb existència obligatòria.

3.3. Data types i exemples.

Els tipus de dades permeten la validació del contingut dels elements i els valors dels atributs. Poden ser tipus simples (simpleType) o tipus complexes (complexType).

Els elements que tenen assignats tipus simples tenen dades de caràcters, però no tenen elements fills o atributs.

Els següents elements tenen tipus simple:

```
<tamany>L<tamany>
<comentari>va molt gran</comentari>
<tamanysDisponibles>10 L 2 M</tamanysDisponibles>
```

Per altra banda, els elements complexos poden tenir elements fills o atributs.

Els següents elements tenen tipus complexe:

```
<tamany sistema="EU">L<tamany>
<comentari>va <b>molt molt</b> gran</comentari>
<tamanysDisponibles>
  <size>10</size>
  <size>2</size>
</tamanysDisponibles>
```

Els atributs sempre tipus simple, doncs els atributs no poden tenir elements fills.

Per declarar l'element <tamanysDisponibles> amb XSD:

```
<xs:element name="tamanysDisponibles">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbound" name="size" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

3.3.1 Tipus complexes

El contingut d'un element són els caràcters de dades i elements fills dintre de les etiquetes. Hi ha **cuatre** tipus de contingut per a tipus complexes: simple, element, mixed i buit. El tipus de contingut és independent dels atributs. És a dir tots els tipus de contingut poden tenir atributs o no.

A continuació tenim un exemple de cadascun i com definir-los a XSD.

```
<!--1 -->
<tamany sistema="EU">10</size>

<!--2 -->
<producte>
  <numero>34D</numero>
  <tamany>10</tamany>
</producte>

<!--3 -->
<carta> Estimat <nomClient>Carles Puig</nomClient> ...</carta>

<!--4 -->
<color valor="blau"/>
```

Definicions amb XSD:

```
<!--1 -->
      <xs:element name="tamany">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="sistema" type="xs:string"></xs:attribute>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>

<!--2 -->
      <xs:element name="tamany">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="numero" type="xs:string"/>
            <xs:element name="tamany" type="xs:integer"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>

<!--3 -->
      <xs:element name="carta">
```

```
      <xs:complexType mixed="true">
        <xs:sequence>
          <xs:element name="nomClient"></xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>

<!--4 -->
    <xs:element name="color">
      <xs:complexType>
        <xs:attribute name="valor" type="TipusColorValors"/>
      </xs:complexType>
    </xs:element>
```

3.4. Indicadors

Hi ha set indicadors a XML. Els indicadors especifiquen com s'utilitzen els elements dintre del document XML.

3.4.1. Indicadors d'ordre

Cada element complexe (excepte els elements buits) conté un únic model de grup.

Sequence L'indicador sequence s'utilitza per indicar l'ordre dels elements (si apareixen) fills de l'element complexe. Els elements contenen un indicador d'ocurrència, com minOccurs, però aquest és opcional.

Exemple:

```
    <xs:element name="employee">
    <xs:complexType >
      <xs:sequence>
        <xs:element name="firstname" type="xs:string"/>
        <xs:element name="lastname" type="xs:string"/>
        <xs:element name="registration-date" type="xs:"/>
      </xs:sequence>
    </xs:complexType>
    </xs:element>
```

All L'indicador all es comporta com el seqüence, excepte que no es necessari que surtin tots els elements ni en el mateix ordre. Si surten, només poden sortir un cop.

```
    <xsd:element name="samarreta">
      <xsd:complexType>
        <xsd:all>
          <xsd:element name="color" type="xsd:string"/>
          <xsd:element name="tamany" type="tamanys-roba"/>
```

```
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

Choice L'indicador choice declara un grup d'elements del qual només sortirà un al document XML

```
<xs:element name="vehicle-motor">
<xs:complexType >
  <xs:sequence>
    <xs:element name="cotxe" type="xs:string"/>
    <xs:element name="moto" type="xs:string"/>
    <xs:element name="camio" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

3.4.2. Indicadors d'ocurrència

Els indicadors d'ocurrència determinen el número de vegades que surt un element en el document XML. Els valors per defecte de minOccurs i maxOccurs és 1.

Els indicadors **maxOccurs** i **minOccurs** es poden utilitzar com a atributs dintre dels xs:element i els indicadors d'ordre (xs:sequence, xs:choice).

Exemples:

```
<xs:element name="book" maxOccurs="unbounded">
<xs:complexType >
  <xs:sequence>
    <xs:element name="cotxe" type="xs:string"/>
    <xs:element name="moto" type="xs:string"/>
    <xs:element name="camio" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

Dintre d'aquests indicadors podem fer servir un valor numèric o **unbounded** que indica un número il.limitat d'elements.

3.4.3. Indicadors de grup

Group name i **attributeGroup name** Podem definir un grup d'elements o atributs, donar-li un nom i fer referència al grup desde una altra definició. Exemple

```
<xs:group name="persona-grup">
  <xs:sequence>
    <xs:element name="nom" type="xs:string"/>
    <xs:element name="cognom" type="xs:string"/>
```

```
    <xs:element name="data-naixement" type="xs:date"/>
  </xs:sequence>
</xs:group>

<xs:element name="person" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:group ref="persona-group"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

3.5. Facetes

Las restriccions a XSD -també anomenades facetes- s'utilizan per a definir un rang de valors acceptable per als elements simples o atributs XML.

1. Restriccions per valors

El següent exemple defineix un element anomenat **edat** amb una restricció. El valor de l'edat no pot ser més petita que 0 o més gran que 120.

```
<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>
```

2. Restriccions per una sèrie de valors

Per limitar el contingunt d'un element XML a una sèrie acceptable de valors, farem servir la restricció d'enumeració. L'exemple que ve a continuació defineix un element anomenat **cotxe** amb una restricció. Els únics valors acceptables son: Audi, Golf, BMW.

```
<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>
```

L'exemple de dalt també es pot escrire com segueix:

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

Nota: En aquest cas el tipus "carType" es pot fer servir en altres elements perquè no és part de l'element "car".

3. Restriccions per una sèrie de valors (patrons)

Per limitar el contingut d'un element XML per a definir una sèrie de números o lletres que es poden fer servir, farem servir la restricció de patró. L'exemple següent defineix un element anomenat "letter" amb una restricció, que consisteix en contenir UNA lletra MINÚSCULA (de a fins a z).

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

En el següent exemple només s'admeten **tres** lletres majúscules (a fins z):

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

En el següent exemple només s'admeten tres lletres **majúscules o minúscules**:

```
<xs:element name="initials">
  <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>
```

En el següent exemple només s'admet **una** de les següents lletres x, y o z:

```
<xs:element name="choice">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[xyz]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Només s'accepten **cinc** digits en seqüència, cada dígit ha d'estar en el rang 0 fins a 9.

```
<xs:element name="prodid">
  <xs:simpleType>
<xs:restriction base="xs:integer">
<xs:pattern value="[0-9][0-9][0-9][0-9][0-9]"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
```

4. Restriccions amb els caràcters "white-space"

Per especificar com es tracten els caràcters "white-space", farem servir la restricció **whiteSpace**. El següent exemple defineix un element anomenat "address" amb una restricció. La restricció está establerta a "preserve", que vol dir que el processador XML no esborrarà cap caracter "white-space" (espais, tabuladors, noves línees i retorns de carro).

```
<xs:element name="address">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="preserve"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Aquest exemple utilitza **replace**, que reemplaça els caràcters **whiteSpace** amb espais:

```
<xs:element name="address">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="replace"/>
    </xs:restriction>
```

```
  </xs:simpleType>
</xs:element>
```

Aquest exemple utilitza **collapse**, que esborra tots els caràcters **whiteSpace**. Els espais en blanc múltiples es redueixen a un únic espai en blanc:

```
<xs:element name="address">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="collapse"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

5. Restriccions de tamany

Per a limitar la longitud d'un valor en un element farem servir les restriccions **length**, **maxLength** i **minLength**.

L'exemple següent defineix un element "password" amb la restricció que el valor ha de ser exactament de vuit caràcters:

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

Aquest exemple defineix altre element "password" amb una restricció. El valor ha de tenir mínim cinc caràcters i un màxim de vuit.

```
<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>
```

6. Restriccions disponibles per als tipus de dades

Restricció	Descripció	Valors
enumeration	Defineix una llista de valors acceptables	
fractionDigits	Número màxim de decimals permesos	>= 0

Restricció	Descripció	Valors
length	Número exacte de caràcters o elements de llista permesos	>= 0
maxExclusive	Limit superior per valors numèrics	<
maxInclusive	Limit superior per valors numèrics	<=
minExclusive	Limit inferior per valors numèrics	>
minInclusive	Limit inferior per valors numèrics	>=
maxLength	Número máxim de caràcters o elements de llista permesos	>= 0
minLength	Número mínim de caràcters o elements de llista permesos	>= 0
pattern	Defineix la seqüència exacta de caràcters permesos	regex
totalDigits	Especifica el número exacte de dígits	>= 0
whiteSpace	Especifica com es tracten els espais en blanc	preserve replace collapse

7. Altres exemples amb restriccions

The example below defines an element called "letter" with a restriction. The acceptable value is zero or more occurrences of lowercase letters from a to z:

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z])*"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

The next example also defines an element called "letter" with a restriction. The acceptable value is one or more pairs of letters, each pair consisting of a lower case letter followed by an upper case letter. For example, "sToP" will be validated by this pattern, but not "Stop" or "STOP" or "stop":

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z][A-Z])+"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

The next example defines an element called "gender" with a restriction. The only acceptable value is male OR female:

```
<xs:element name="gender">
```

```
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="male|female"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

The next example defines an element called "password" with a restriction. There must be exactly eight characters in a row and those characters must be lowercase or uppercase letters from a to z, or a number from 0 to 9:

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z0-9]{8}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```