Llenguatges de Marques i Sistemes de Gestió d'Informació

UD 2.3 Validació XML

IES Jaume II El Just
Tavernes de la Valldigna
Departament d'Informàtica
Curs 2024-25

Índex

1 Validació d'XML **2**
1.1 DTD . . . 2
1.1.1 Declaració del DTD . . . 2
1.1.2 Declaració de Tipus d'Elements . . . 3
1.1.3 Declaració de Tipus d'Atributs . . . 4

2 Espais de noms **5**
2.1 Resolució del conflicte de noms mitjançant un prefix . . . 5

3 XML Schema **8**
3.1 Declaració de l'esquema . . . 8
3.2 Definició d'esquemes . . . 8
3.3 Elements de XSD . . . 9
3.3.1 Elements simples . . . 9
3.3.2 Atributs . . . 10
3.3.3 Restriccions . . . 11
3.3.4 Elements complexos . . . 13
3.3.5 Indicadors . . . 16
3.4 Exemples . . . 18
3.4.1 Nota: . . . 18
3.4.2 Mensaje: . . . 19

4 Referències **19**

1 Validació d'XML

Per a saber si un document XML és vàlid, hem de comprovar si està ben format i si valida contra un esquema determinat. Els llenguatges per a definir esquemes més utilitzats són **Document Type Definition (DTD)**, **XML Schema Definition(XSD)** i **Relax NG**.

1.1 DTD

El fitxer DTD conté la definició d'elements, atributs i entitats permeses, i pot expressar algunes limitacions per combinar-los. Su'utilitza en fitxers SGML i XML.

El primer que s'ha de verificar és que el document XML està ben format (*well formed*), seguint les regles vistes a l'apartat anterior.

Després podem validar el document XML utilitzant un *parser* que aplica les definicions del fitxer DTD. Un document XML vàlid és aquell que no només està ben format, sinó que també segueix les regles del DTD.

1.1.1 Declaració del DTD

Per a declarar quin DTD hem d'utilitzar tenim 3 opcions:

1.1.1.1 Declaració interna en el mateix document XML

Exemple de document XML amb DTD inclòs.

Exemple:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE nota [
  <!ELEMENT nota (destinatari, remitent, capcalera, cos)>
  <!ELEMENT destinatari (#PCDATA)>
  <!ELEMENT remitent (#PCDATA)>
  <!ELEMENT capcalera (#PCDATA)>
  <!ELEMENT cos (#PCDATA)>
]>

<nota>
  <destinatari>Tove</destinatari>
  <remitent>Jani</remitent>
  <capcalera>Recordatori</capcalera>
```

```
  <cos>Truca'm!</cos>
</nota>
```

1.1.1.2 Declaració en un fitxer extern local

```
<!DOCTYPE element_arrel SYSTEM "document.dtd">
```

1.1.1.3 Declaració en un fitxer extern public

```
<!DOCTYPE element_arrel PUBLIC "url/document.dtd">
```

Exemple de referència pública:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
↪  "http://www.w3.org/TR/html4/strict.dtd">
```

Aquest DTD s'utilitza per validar documents XHTML estrictes.

1.1.2 Declaració de Tipus d'Elements

Sintaxi:

```
<!ELEMENT nom_element tipus_de_contingut>
```

El nom ha de ser vàlid a XML i únic.

Tipus d'elements:

- Només contenen elements: s'especifica entre parèntesis l'identificador de cada sub-element.
  - Relació sequencial: es separa cada element amb comes:
    `<!ELEMENT identificació (element1, element2, ...) >`
  - Relació alternativa: es separa cada element amb |
    `<!ELEMENT nom (element1 | element2)>`

També podem indicar la freqüència de repetició de cada element.

> Indicadors de freqüència: * *: 0 o més vegades. * ?: 0 o 1 vegada. * +: 1 o més vegades.

Per exemple:

```
<!ELEMENT identificacio (situacio?,nom_complet, nick*,mail?)>
```

- Només contenen dades (PCDATA):
  `<!ELEMENT nom (#PCDATA)>`

- Elements buits:
  `<!ELEMENT nom EMPTY>`
- Elements mixtos (PCDATA + elements):
  Ex.: `<!ELEMENT comentari (#PCDATA | elem1 | elem2)*>`

1.1.3 Declaració de Tipus d'Atributs

Sintaxi: `<!ATTLIST nom_element nom_atribut tipus predeterminacio>`

Tipus d'atributs:

- **CDATA**, **NMTOKEN**, **NMTOKENS**:
  - **CDATA**: Qualsevol text.
  - **NMTOKEN**: Valors sense espais i només lletres, números, subratllats.
  - **NMTOKENS**: Llista de NMTOKENs.
- Atributs enumerats:
  Ex.: `<!ATTLIST cotxe color (blanc | negre | gris) "negre">`
- **ID**, **IDREF**, **IDREFS**:
  - **ID**: Un valor únic per a identificar elements.
  - **IDREF**: Referència a un ID d'un altre element.
  - **IDREFS**: Llista de referències a ID d'altres elements.

Predeterminació d'Atributs:

- **#REQUIRED**: Atribut obligatori.
- **#IMPLIED**: Atribut opcional.
- **#FIXED**: Valor fixat declarat en el DTD
- Valor per defecte: Definició predeterminada.

Exemples:

```
<!ATTLIST cotxe color CDATA #IMPLIED>
<!ATTLIST cotxe matricula ID #REQUIRED>
<!ATTLIST cotxe marca CDATA FIXED "Seat">
<!ATTLIST cotxe color CDATA "vermell">
```

2 Espais de noms

XML **Namespaces** (espais de noms) són un mètode per a evitar els conflictes de noms.

Per exemple, si usem documents XML de diferents fonts amb el mateix nom en elements distints, es produeix un conflicte de noms.

Per exemple, podem tindre un document XML amb una etiqueta *persona* definida així:

```
<persona>
  <nom> ... </nom>
  <cognoms> .. </cognoms>
</persona>
```

i un altre document on es defineix així:

```
<persona>
  <nom> ... </nom>
  <cognom1> .. </cognom1>
  <cognom2> .. </cognom2>
</persona>
```

Si afegim aquests fragments XML junts, hi hauria un conflicte de noms. Tots dos contenen un element `<persona>`, però tenen contingut diferent.

2.1 Resolució del conflicte de noms mitjançant un prefix

Els conflictes de noms en XML es poden evitar fàcilment mitjançant un **prefix de nom**.

El problema anterior es pot resoldre així:

```
<client:persona>
  <client:nom> ... </client:nom>
  <client:cognoms> .. </client:cognoms>
</client:persona>

<alumne:persona>
  <alumne:nom> ... </alumne:nom>
  <alumne:cognom1> .. </alumne:cognom1>
  <alumne:cognom2> .. </alumne:cognom2>
</alumne:persona>
```

Quan s'utilitzen prefixos en XML, s'ha de definir un espai de noms per al prefix.

L'espai de noms es pot indicar amb un atribut **xmlns** a l'etiqueta inicial d'un element i té la sintaxi següent:

```
xmlns:prefix="URI"
```

> L'**identificador uniforme de recursos** (**Uniform Resource Identifier**, **URI**) és un text curt que identifica sense equivocació qualsevol recurs (servei, pàgina, document, adreça de correu electrònic ...) accessible en una xarxa. També engloba els **URL**, acrònim de **Uniform Resource Locator**.

```
<root>
  <client:persona xmlns:client="namespace/client">
    <client:nom> ... </client:nom>
    <client:cognoms> .. </client:cognoms>
  </client:persona>

  <alumne:persona xmlns:alumne="http://example.org/alumne">
    <alumne:nom> ... </alumne:nom>
    <alumne:cognom1> .. </alumne:cognom1>
    <alumne:cognom2> .. </alumne:cognom2>
  </alumne:persona>
</root>
```

A l'exemple anterior:

L'atribut `xmlns` del primer element `<persona>` dóna al prefix `client:` un espai de noms qualificat. L'atribut `xmlns` del segon element `<persona>` dóna al prefix `alumne:` un espai de noms qualificat.

Quan es defineix un espai de noms per a un element, tots els elements fills amb el mateix prefix s'associen amb el mateix espai de noms.

L'URI de l'espai de noms no és utilitzat per l'analitzador. El seu propòsit és donar a l'espai de noms un nom únic. Les empreses sovint utilitzen l'espai de noms com a punter a una pàgina web que conté informació sobre l'espai de noms.

L'exemple anterior també es pot fer declarant l'espai de noms a l'element arrel XML:

```
<root xmlns:client="namespace/client"
↪   xmlns:alumne="http://example.org/alumne">
  <client:persona>
    <client:nom> ... </client:nom>
    <client:cognoms> .. </client:cognoms>
  </client:persona>
```

```
  <alumne:persona>
    <alumne:nom> ... </alumne:nom>
    <alumne:cognom1> .. </alumne:cognom1>
    <alumne:cognom2> .. </alumne:cognom2>
  </alumne:persona>
</root>
```

Quan es defineix un namespace per a un element, els seus descendents també l'hereten i no és necessari indicar-ho en ells:

```
<persona xmlns="namespace/client">
    <nom> ... </nom>
    <cognoms> .. </cognoms>
</persona>
```

3 XML Schema

Una altra forma alternativa a DTD de validar documents és amb un llenguatge anomenat **XML Schema** (també anomenat **XSD** o **XML Schema Definition**).

Es pot dir que un XSD (o esquema) és l'evolució natural d'un DTD. Permet expressar amb més potència, gramàtiques més complexes utilitzant la mateixa sintaxi XML. Va néixer el 1998 i es va recomanar l'ús el 2001 pel W3C (World Wide Web Consortium).

El principal avantatge de XSD és que en estar basat en XML, és fàcilment extensible a les futures modificacions o necessitats.

També permet definir de manera molt clara els tipus de dades i els espais de noms que se suporten. Això és realment important ja que si un element se sap que és de tipus *double* i, finalment, s'escriu com a string o integer, no s'admetrà la validesa del document.

És més, es poden definir els tipus de dades exactament igual que si fora una base de dades, facilitant la conversió d'un a un altre i les transferències informació.

3.1 Declaració de l'esquema

Per a indicar quin esquema utilitza un document XML s'escriu aquest en l'element arrel:

```
<root xmlns="https://example.org"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://example.org/ document.xsd">
```

L'exemple anterior és per a un esquema públic en Internet, indicant que el document de l'esquema és `document.xsd` i es troba en l'URL `http://example.org/`.

Si l'esquema és local:

```
<root xmlns="https://example.org"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="local_schema.xsd">
```

O sense espai de noms:

```
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="local_schema.xsd">
```

3.2 Definició d'esquemes

L'element `<schema>` és l'element arrel de cada esquema XML:

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
...
...
</xs:schema>
```

L'atribut `xmlns:xs="http://www.w3.org/2001/XMLSchema"` indica que els elements i els tipus de dades utilitzats a l'esquema provenen de l'espai de noms `"http://www.w3.org/2001/XMLSchema"`. També especifica que els elements i els tipus de dades que provenen de l'espai de noms `"http://www.w3.org/2001/XMLSchema"` han de tenir el prefix **`xs:`**.

Una capçalera més completa per a xs:schema seria:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://example.org"
xmlns="http://example.org">
```

`targetNamespace` indica que els elements definits per aquest esquema provenen de l'espai de noms `"http://example.org"` i **`xmlns`** indica que l'espai de noms predeterminat és `"http://example.org"`.

3.3 Elements de XSD

Els **elements** de XSD Schema poden ser simples o complexos.

Els **elements simples** només poden contindre text, no poden contindre altres elements o atributs.

Els **elements complexos** poden contindre altres elements i/o atributs.

3.3.1 Elements simples

El text dels elements simples pot ser de molts tipus diferents. Pot ser un dels tipus inclosos a la definició de l'esquema XML (booleà, string, etc.), o pot ser un tipus personalitzat que podem definir nosaltres.

També podem afegir restriccions (*facets*) a un tipus de dades per limitar-ne el contingut, o bé podem demanar que les dades coincideixin amb un patró específic.

La sintaxi per a definir un element simple és:

```
<xs:element name="nom_element" type="tipus_element"/>
```

On `name` és el nom de l'element i `type` és el tipus.

Els tipus més usats són:

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

Per exemple, el fragment XML:

```
<nom>Andreu</nom>
<edat>23</edat>
<data_naixement>2019-06-25</data_naixement>
```

tindria el següent esquema:

```
<xs:element name="nom" type="xs:string"/>
<xs:element name="edat" type="xs:integer"/>
<xs:element name="data_naixement" type="xs:date"/>
```

Els elements simples poden tindre un **valor per defecte** o un **valor fixe**.

S'assigna automàticament un **valor per defecte** a l'element quan no s'especifica cap altre valor.

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

Si indiquem que un **valor es fixe**, sempre té eixe valor i no pot tindre un altre:

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

3.3.2 Atributs

Els atributs es declaren com a **tipus simples**. Els elements simples no poden tindre atributs. Si un element té atributs, es considera que és de tipus complex, però l'atribut en si es declara sempre com un tipus simple.

La sintaxi per a definir un atribut és:

```
<xs:attribute name="nom_atribut" type="tipus_atribut"/>
```

Exemple:

```
<article lang="es">
```

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

Els atributs poden tindre els mateixos tipus que els elements. També poden tindre un valor `default` o `fixed`.

A més, també poden ser **opcionals** o **requerits**. Si no s'especifica res, sempre són opcionals (`use="optional"`). Per a indicar que són requerits s'utilitza `use="required"`:

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

Si s'especifica que un atribut té un **valor per defecte** (`default="valor"`), no és necessari indicar que és requerit.

```
<xs:attribute name="lang" type="xs:string" default="en" />
```

3.3.3 Restriccions

Quan un element o atribut XML té un tipus de dades definit, imposa restriccions al contingut de l'element o atribut.

Si un element XML és de tipus `"xs:integer"` i conté una cadena com `"Hola"`, l'element no es validarà.

A més, amb XSD, també podem afegir les nostres pròpies restriccions als elements i atributs. Aquestes restriccions s'anomenen *facets*.

Per exemple, el següent defineix un element *edat* que ha de ser un número enter entre 0 i 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>
```

En aquesta estructura es pot veure les etiquetes que es necessiten per a definir la restricció:

- `<xs:element name="nom_element">` : defineix el nom de l'element.
- `<xs:simpleType>` : indica que és un tipus simple.
- `<xs:restriction base="xs:tipus">` : indica el tipus de l'element.

La mateixa definició es podria escriure així:

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

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

Aquesta segona forma és millor si el tipus `tipus_edat` pot ser reutilitzat per altres elements.

Per a limitar el contingut d'un element XML a un **conjunt de valors** acceptables, utilitzem la restricció d'enumeració.

L'exemple següent defineix un element anomenat `"color"` amb una restricció. Els únics valors acceptables són: *negre*, *blanc* i *roig*:

```
<xs:element name="color">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="negre"/>
      <xs:enumeration value="blanc"/>
      <xs:enumeration value="roig"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

El mateix esquema seria aplicable als atributs. Per exemple, un atribut que només pot tindre els valor "blanc" o "negre" i el valor per defecte és "blanc":

```
<xs:attribute name="color" default="blanc">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="blanc" />
            <xs:enumeration value="negre" />
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
```

Si no volem restringir els valor sinó estendre'ls, hem d'utilitzar **extension** en lloc de **restriction**. Per exemple, el següent codi validaria un element anomenat *plaza* de tipus string amb un atribut booleà (*estaLibre*) que per defecte és **false**:

```
<xs:element name="plaza">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="estaLibre" type="xs:boolean"
↪  default="false" />
            </xs:extension>
↪
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

També es poden usar expressions regulars per a especificar patrons. alguns exemples:

```
<xs:restriction base="xs:string">
  <!-- 0 o més lletres minúscules -->
  <xs:pattern value="([a-z])*" />
</xs:restriction>

<xs:restriction base="xs:string">
  <!-- Només els valors "on" i "off" -->
  <xs:pattern value="on|off" />
</xs:restriction>
```

També es pot limitar la longitud d'una cadena de caracters amb `xs:length`:

```
<xs:restriction base="xs:string">
  <!-- Màxim 10 caracters -->
  <xs:length value="10"/>
</xs:restriction>
```

Mes sobre restriccions.

3.3.4 Elements complexos

Un **element complex** és un element XML que conté altres elements i/o atributs.

Hi ha quatre tipus d'elements complexos:

- elements buits

- elements que només contenen altres elements
- elements que només contenen text
- elements que contenen tant altres elements com text

Cadascun d'aquests elements també pot contenir atributs.

3.3.4.1 Elements complexos que contenen només altres elements

S'utilitza `<xs:sequence>` per a definir elements que només contenen altres elements:

```
<xs:element name="alumne" type="tipus_persona"/>

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

En aquest exemple `<nom>` i `<cognom>` han d'aparèixer obligatòriament i en el mateix ordre:

```
<!-- Correcte -->
<alumne>
  <nom>Maria</nom>
  <cognom>Llopis</cognom>
</alumne>

<!-- Incorrecte -->
<alumne>
  <cognom>Llopis</cognom>
  <nom>Maria</nom>
</alumne>

<!-- Incorrecte -->
<alumne>
  <nom>Maria</nom>
</alumne>
```

3.3.4.2 Elements complexos buits (empty) Hi ha diferents formes de definir un element que no pot contindre altres elements:

```
<xs:element name="element_buit">
    <xs:complexType/>
</xs:element>
```

```
<element_buit />
```

Si volem que tinga atributs:

```
<xs:element name="producte">
  <xs:complexType>
    <xs:attribute name="id" type="xs:integer"/>
  </xs:complexType>
</xs:element>
```

```
<producte id="223" />
```

3.3.4.3 Elements complexos amb només text La diferència entre aquest tipus d'element i els elements simples és que els simples no poden tindre text i atributs. Si volem que un element tinga text i atributs, hem de declarar-lo com a complex.

Aquest tipus només conté contingut simple (text i atributs), per tant, afegim un element `simpleContent` al voltant del contingut. A més, hem de definir una **extensió** O una **restricció** dins de l'element `simpleContent`, com per exemple:

```
<xs:element name="preu">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:decimal">
        <xs:attribute name="moneda" type="xs:string" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

validaria el següent fragment XML:

```
<preu moneda="euro">25</preu>
```

3.3.4.4 Elements complexos mixtes Els elements mixtes poden contindre text i altres elements. S'especifiquen amb l'atribut `mixed="true"`:

```
<xs:element name="missatge">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="destinatari" type="xs:string"/>
      <xs:element name="data" type="xs:date"/>
      <xs:element name="cos" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<missatge>
  Per a: <destinatari>John Smith</destinatari>.
  Amb data: <data>2023-01-10</data>
  Cos del missatge: <cos>Hola, com va?</cos>.
</missatge>
```

En aquest exemple, `<destinatari>`, `<data>` i `<cos>` han d'aparèixer sempre i en el mateix ordre, pero poden tindre text intercalat.

3.3.5 Indicadors

Els indicadors serveixen per indicar com apareixen els elements XML

3.3.5.1 Indicadors d'ordre L'indicador **`xs:all`** especifica que els elements poden aparèixer en qualsevol ordre i sempre una vegada:

```
<xs:element name="persona">
  <xs:complexType>
    <xs:all>
      <xs:element name="altura" type="xs:decimal"/>
      <xs:element name="pes" type="xs:integerS"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

Amb l'exemple anterior serien vàlides:

```
<persona>
  <altura>1.70</altura>
  <pes>75</pes>
<persona>
```

i

```
<persona>
  <pes>75</pes>
  <altura>1.70</altura>
<persona>
```

L'indicador **`xs:choice`** especifica que, de tots els elements fills, només pot aparèixer un:

```
<xs:element name="persona">
  <xs:complexType>
    <xs:choice>
      <xs:element name="alumne" type="tipus_alumne"/>
      <xs:element name="professor" type="tipus_professor"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

L'indicador **`xs:sequence`** ja el coneixem. Especifica que tots els elements fills han d'aparèixer en el mateix ordre en el que estan especificats.

3.3.5.2 Indicadors d'ocurrència

Els indicadors d'ocurrència són **`minOccurs`** i **`maxOccurs`**, i especifiquen el nombre mínim i màxim d'ocurrències d'un element.

```
<xs:element name="alumne">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="assignatura" type="xs:string" minOccurs="2"
↪   maxOccurs="10"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

El valor per defecte d'ambdos indicadors és `"1"`.

Per a permetre infinites ocurrències d'un element, s'usa el valor `maxOccurs="unbounded"`.

Equivalències entre les multiplicitats de DTD i els indicadors d'ocurrència de XSD:

Multiplicitat	DTD	XSD
0 o 1	`?`	`minOccurs="0"`
0 o varis	`*`	`minOccurs="0" maxOccurs="unbounded"`
1 o varis	`+`	`maxOccurs="unbounded"`

3.4 Exemples

3.4.1 Nota:

XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<nota xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="nota.xsd">
    <destinatario>Pep</destinatario>
    <remitente>Jenny</remitente>
    <cabecera>Recordatorio</cabecera>
    <cuerpo>Llámame!</cuerpo>
</nota>
```

XSD

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="nota">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="destinatario" type="xs:string"/>
                <xs:element name="remitente" type="xs:string"/>
                <xs:element name="cabecera" type="xs:string"/>
                <xs:element name="cuerpo" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

3.4.2 Mensaje:

XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<mensaje xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mensaje.xsd"
prioridad="urgente">
    <de>Alfredo Reino</de>
    <a>Hans van Parijs</a>
    <texto idioma="holandés"> Hallo Hans, hoe gaat het?
    </texto>
</mensaje>
```

XSD

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="mensaje">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="de" type="xs:string" />
        <xs:element name="a" type="xs:string" />
        <xs:element name="texto">
            <xs:complexType>
                <xs:simpleContent>
                    <xs:extension base="xs:string">
                        <xs:attribute name="idioma" type="xs:string" />
                    </xs:extension>
                </xs:simpleContent>
            </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="prioridad" type="xs:string" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

4 Referències

- W3Schools XSD
- Especificació de XSD