UD2-Definicion de esquemas y vocabulario XSD

Contenido

1. Introducción

Los XML Schema son ficheros en texto plano con la **extensión XSD** que describen la estructura de un archivo XML.

El nodo raíz de un XSD es **<xs:schema>** y contiene todos los elementos y atributos del documento XML asociado.

1.1 Como asociar un documento XSD a un documento XML

El modo de asociar un esquema a un documento XML es añadiendo la siguiente línea al nodo raíz del documento. Donde indicaremos el nombre del archivo XSD.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<alumnos xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="alumnos.xsd">
  <alumno>
    <nombre>José Ramón</nombre>
    <apellidos>García González</apellidos>
  </alumno>
  <alumno>
    <nombre>Carlos</nombre>
    <apellidos>López Pérez</apellidos>
  </alumno>
</alumnos>
```

Para vincular un esquema a un documento XML, es obligatorio que este último haga referencia al espacio de nombres http://www.w3.org/2001/XMLSchema-instance. Para ello, habitualmente se utiliza el prefijo xsi, aunque se puede utilizar otros como xs.

El atributo **noNameSchemaLocation** permite referenciar a un archivo con la definición de un esquema que no tiene ningún espacio de nombres asociado. En este caso, dicho archivo es “alumnos.xsd”.

El esquema XML guardado en **alumnos.xsd** y que permita validar el documento XML alumnos.xml podría ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="alumnos">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="alumno" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="nombre" type="xs:string"/>
       <xs:element name="apellidos" type="xs:string"/>
       </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Para estar bien formado, un esquema XML tiene que cumplir las mismas reglas de sintaxis que cualquier otro documento XML.

Por otra parte, hay que tener en cuenta que, en todos los esquemas XML, el elemento raíz es “schema”. Ahora bien, para escribirlo, es muy común utilizar el prefijo xsd o xs.

Con xmlns:xs="http://www.w3.org/2001/XMLSchema" se ha indicado que:

- Los elementos y tipos de datos utilizados en el esquema pertenecen al espacio de nombres http://www.w3.org/2001/XMLSchema.
- Dichos elementos y tipos de datos deben llevar el prefijo xs (xs:schema, xs:element, xs:complexType, xs:string...).

Fíjese también que:

- Los elementos “marcadores” y “página” son de tipo complejo (complexType), ya que, contienen a otros elementos.
- **sequence** indica que los elementos hijo deben aparecer, en el documento XML, en el mismo orden en el que sean declarados en el esquema.
- Los elementos “nombre”, “descripción” y “url” son de tipo simple (string en este caso) y no pueden contener a otros elementos.
- Mediante **maxOccurs="unbounded**" se ha indicado que pueden aparecer ilimitados elementos “página” en el documento XML.

A lo largo de este documento se detalla y amplia esta información.

2. Tipos de elementos

A la hora de escribir xml schema o XSD podemos distinguir entre dos tipos de **elementos simples y los complejos**. Veamos el siguiente ejemplo:

```
<alumnos>
  <alumno>
    <nombre>José Ramón</nombre>
    <apellidos>García González</apellidos>
  </alumno>
  <alumno>
    <nombre>Carlos</nombre>
    <apellidos>López Pérez</apellidos>
  </alumno>
</alumnos>
```

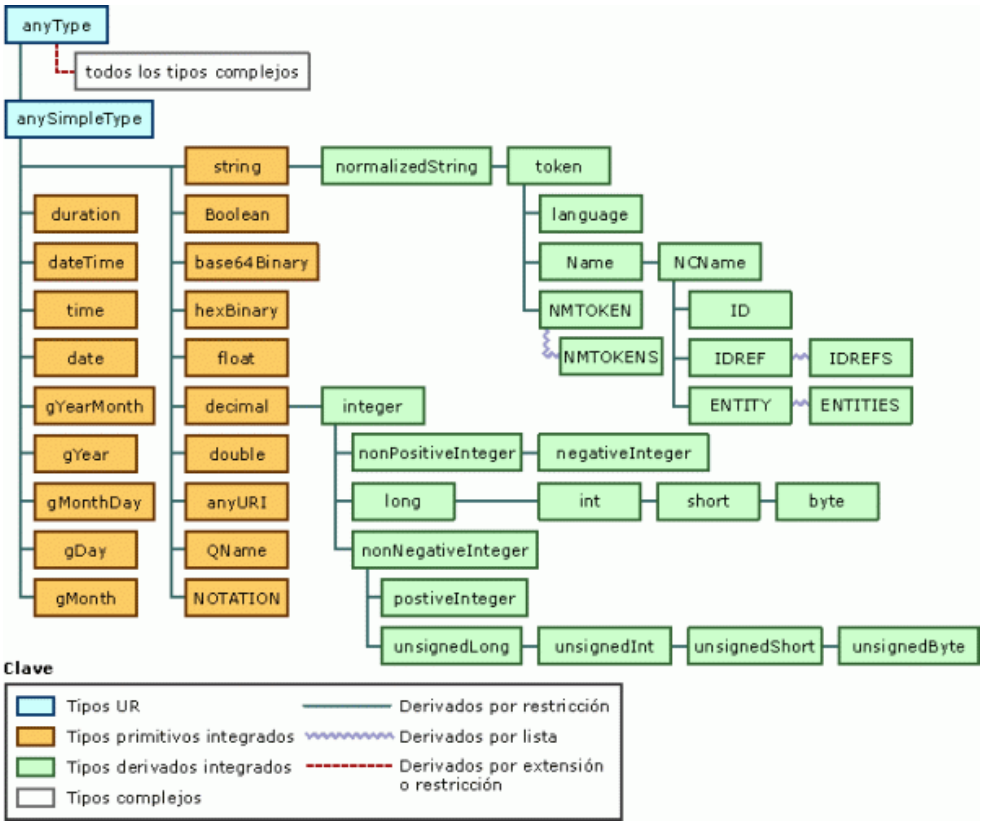
En este caso los **elementos simples son las etiquetas <nombre> y <apellidos>** ya que solamente pueden contener texto.

Los elementos complejos son <alumnos> y <alumno> porque contienen otros elementos.

3. Elementos simples

Los elementos simples solamente pueden contener texto y no pueden contener a otro u otros elementos (hijos), ni tampoco pueden tener atributos.

El texto contenido en un elemento simple puede ser de diferentes tipos: Primitivos (string, boolean, decimal...) o derivados de estos (integer, ID, IDREF...). Véase la siguiente imagen donde se puede ver la relación que existe entre todos ellos:



Para definir un elemento simple se puede utilizar la siguiente sintaxis:

```
<xs:element name="xxx" type="yyy"/>
```

Ejemplo:

```
<lastname>Refsnes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

En los elementos simples también podemos definir las declaraciones **fixed** y **default**. Fixed para indicar que el valor es fijo, y default para especificar un valor por defecto en caso de que no se especifique uno.

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

Cabe destacar los siguientes aspectos que abordaremos a lo largo del documento:

- Un elemento vacío y sin atributos en XSD requiere una explicación a parte ya que puede definirse como elemento simple o complejo.
- Aunque hemos visto que la declaración de los elementos simples de forma compacta (ocupando solo una línea), su declaración se puede extender para matizar su definición.

De esta forma solo definimos el elemento **<notamedia>** que va a contener un numero entero.

```
<xs:element name="notamedia" type="xs:integer">
```

De esta forma definimos el elemento **<notamedia>** que tendrá que cumplir con determinadas restricciones:

- Primero que debe ser un numero entero.
- Su valor mínimo será 0 y su valor máximo será 10.

```
<xs:element name="notamedia">
  <xs:simpleType>
   <xs:restriction base="xs:integer">
     <xs:minInclusive value="0"/>
     <xs:maxInclusive value="10"/>
   </xs:restriction>
  </xs:simpleType>
</xs:element>
```

4. Elementos complejos

Un elemento es complejo cuando contiene uno o más elementos (hijos) y/o atributos. Por ejemplo:

```
<employee>
 <firstname>John</firstname>
 <lastname>Smith</lastname>
</employee>
```

```
<employee id="1">John Smith</employee>
```

```
<employee id="1"/>
```

4.1 Como declarar un elemento complejo

Partiendo del siguiente ejemplo:

```
<employee>
    <firstname>John</firstname>
    <lastname>Smith</lastname>
</employee>
```

La forma de declarar un elemento complejo sería la siguiente:

- Nótese que utilizamos **<xs:complexType>** para definir que se trata de un elemento complejo.
- La etiqueta **<xs:sequence>** indica los elementos hijos que tendrá este elemento complejo. (Existen otros dos tipos de etiquetas que podemos utilizar que se verán mas adelante).

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

4.2 Cardinalidad o repetición de elementos.

Este indicador se utiliza para indicar el número de veces que un elemento hijo se repetirá y se define a través de los atributos:

- **minOccurs** – indica el número minino de veces que aparecerá el elemento.
- **maxOccurs**. – indica el número máximo de veces que puede aparecer dicho elemento.

Tomemos el siguiente ejemplo:

```xml
<?xml version = "1.0" encoding = "UTF-8"?>
<Movies xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="ejemplo1.xsd">
    <Movie>
        <Title>The Godfather</Title>
        <Genre>Crime Drama </Genre>
        <Studio>Paramount Pictures </Studio>
        <Year>1972</Year>
    </Movie>
    <Movie>
        <Title>The Shwashank Redemption</Title>
        <Genre>Drama</Genre>
        <Studio>Columbia Pictures </Studio>
        <Year>1994</Year>
    </Movie>
</Movies>
```

Analizamos el archivo XSD que define su estructura:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Movies">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Movie" minOccurs="0" maxOccurs="100">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Title" type="xs:string"/>
                            <xs:element name="Genre" type="xs:string"/>
                            <xs:element name="Studio" type="xs:string"/>
                            <xs:element name="Year" type="xs:gYear"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

En la línea resaltada veremos que estamos permitiendo que el elemento <Movie> pueda no aparecer ninguna vez, o 100 veces.

- Si quisiéramos que fuera un número ilimitado o indefinido de veces **maxOccurs="unbounded".**
- En caso de no indicarlos minOccurs y maxOccurs toman por defecto el valor 1.
- Si quisiéramos que un elemento fuera opcional utilizaríamos minOccurs="0" y maxOccurs="1".

Más información: https://www.w3.org/TR/xmlschema-0/#OccurrenceConstraints

4.3 Indicador de orden

Los elementos complejos pueden indicar de 3 formas distintas como son sus nodos hijos:

Indicador SEQUENCE

Indica que los elementos hijo deben aparecer en el mismo orden que se declaran.

```
<xs:element name="persona">
  <xs:complexType>
   <xs:sequence>
     <xs:element name="nombre" type="xs:string"/>
     <xs:element name="apellido" type="xs:string"/>
   </xs:sequence>
 </xs:complexType>
</xs:element>
```

Indicador ALL

Indica que los elementos hijo pueden aparecer en cualquier orden, pero que solo pueden aparecer una vez cada uno. (Aunque en estructuras complejas se puede repetir elementos)

```
<xs:element name="person">
 <xs:complexType>
  <xs:all>
   <xs:element name="firstname" type="xs:string"/>
   <xs:element name="lastname" type="xs:string"/>
  </xs:all>
 </xs:complexType>
</xs:element>
```

Indicador CHOICE

Indica que solo puede aparecer uno de los siguientes elementos hijos:

```
<xs:element name="vehiculo">
 <xs:complexType>
  <xs:choice>
   <xs:element name="coche" type="xs:string "/>
   <xs:element name="moto" type="xs:string "/>
   <xs:element name="camion" type="xs:string "/>
  </xs:choice>
 </xs:complexType>
</xs:element>
```

5. Crear tipos de elementos

Cuando trabajamos con estructuras XML complejas podemos acabar escribiendo archivos muy "anchos" conforme vayamos declarando hijos e indentemos o "tabulemos" el código.

También se puede dar la situación en que necesitemos reutilizar estructuras de código. En el siguiente ejemplo **<alumno>** y **<profesor>** comparten los elementos.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<colegio>
    <alumno>
        <nombre>Antonio</nombre>
        <apellido>Perez</apellido>
    </alumno>
    <profesor>
        <nombre>Marisa</nombre>
        <apellido>Gutierrez</apellido>
    </profesor>
</colegio>
```

La solución que XSD nos proporciona es definir tipos de elementos que más adelante podremos reutilizar.

A continuación esta sería la solución convencional, sin utilizar ni definir tipos de elementos:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="colegio">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="alumno" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="nombre" type="xs:string"/>
                            <xs:element name="apellido" type="xs:string"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="profesor" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="nombre" type="xs:string"/>
                            <xs:element name="apellido" type="xs:string"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Esta es la solución definiendo el tipo de elemento persona que luego utiliza los elementos alumno y profesor.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="colegio">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="alumno" type="persona" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="profesor" type="persona" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="persona">
        <xs:sequence>
            <xs:element name="nombre" type="xs:string"/>
            <xs:element name="apellido" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

5.1 Extender la información de un tipo de elemento

En caso que no todos los campos fueran comunes podríamos crear un tipo de elemento base con los elemento comunes y luego extenderlo. Vease el siguiente ejemplo donde alumno y profesor tienen un campo no común.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<colegio  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="ejemplo1.xsd">
    <alumno>
        <nombre>Antonio</nombre>
        <apellido>Perez</apellido>
        <notamedia>9</notamedia>
    </alumno>
    <profesor>
        <nombre>Marisa</nombre>
        <apellido>Gutierrez</apellido>
        <codigo>13</codigo>
    </profesor>
</colegio>
```

A continuación se muestra la solución creando un elemento base persona que luego extienden el tipo profesor y el tipo alumno.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="colegio">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="alumno" type="alumno" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="profesor" type="profesor" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="persona">
        <xs:sequence>
            <xs:element name="nombre" type="xs:string"/>
            <xs:element name="apellido" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="profesor">
        <xs:complexContent>
            <xs:extension base="persona">
                <xs:sequence>
                    <xs:element name="codigo" type="xs:string"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="alumno">
        <xs:complexContent>
            <xs:extension base="persona">
                <xs:sequence>
                    <xs:element name="notamedia" type="xs:integer"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

</xs:schema>
```

6. Atributos

Para definir un atributo se puede emplear la siguiente sintaxis:

```
<xs:attribute name="nombre atributo" type="tipo de dato"/>
```

- Todos los atributos pueden tomar por valor tipos simples.
- Cuando un elemento tiene al menos un atributo dicho elemento se dice que es complejo.

Ejemplo.

```
<?xml version="1.0" encoding="UTF-8"?>
<colegio  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="ejemplo1.xsd">
    <alumno id="alud01a">
        <nombre>Antonio</nombre>
        <apellidos>Perez</apellidos>
    </alumno>
</colegio>
```

En archivo XSD correspondiente seria:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="colegio">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="alumno" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="nombre" type="xs:string"/>
                            <xs:element name="apellidos" type="xs:string"/>
                        </xs:sequence>
                        <xs:attribute name="id" type="xs:ID" use="required"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

En los atributos podemos definir valores por defecto cuando no se ha especificado otro valor:

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

Asignar automáticamente un valor, y no puede ser especificado otro distinto:

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

Los atributos son opcionales por defecto, salvo que se indique mediante el atributo use=”required” que son obligatorios.

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

7. Restricciones

XML Schema permite definir restricciones a los posibles valores de los tipos de datos. Lo que nos permite definir restricciones sobre los posibles valores de atributos o elementos. A continuación tenemos una lista de posibles restricciones:

Faceta	*Descripción*
xs:length	Especifica una longitud fija.
xs:minLength	Especifica una longitud mínima.
xs:maxLength	Especifica una longitud máxima.
xs:pattern	Especifica un patrón de caracteres admitidos.
xs:enumeration	Especifica una lista de valores admitidos.
xs:whiteSpace	Especifica cómo se debe tratar a los posibles espacios en blanco, las tabulaciones, los saltos de línea y los retornos de carro que puedan aparecer.
xs:maxInclusive	Especifica que el valor debe ser menor o igual que el indicado.
xs:maxExclusive	Especifica que el valor debe ser menor que el indicado.
xs:minExclusive	Especifica que el valor debe ser mayor que el indicado.
xs:minInclusive	Especifica que el valor debe ser mayor o igual que el indicado.
xs:totalDigits	Especifica el número máximo de dígitos que puede tener un número.
xs:fractionDigits	Especifica el número máximo de decimales que puede tener un número.

7.1 xs:minExclusive y xs:maxInclusive

```
<xs:element name="nota">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="10"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

En el siguiente código se define un elemento llamado con la restricción de que el valor que tome no pueda ser menor que 0 ni mayor que 10:

- **xs:simpleType** permite definir un tipo simple y especificar sus restricciones.
- **xs:restriction** sirve para definir una restricción y necesita especificar una base que indica el tipo de dato a partir del cual se define la restricción.
- **xs:minInclusive** sirve para especificar que el valor debe ser mayor o igual que el indicado en su atributo value, (en este caso, mayor o igual que 0).
- **xs:maxInclusive** sirve para especificar que el valor debe ser menor o igual que el indicado en su atributo value, (en este caso, menor o igual que 10).

7.2 xs:enumeration

Sirve para definir una lista de valores admitidos. En el siguiente ejemplo se define un elemento llamado “color” con la restricción de que los únicos valores admitidos son: "verde", "amarillo" y "rojo".

```
<xs:element name="color">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="verde"/>
            <xs:enumeration value="amarillo"/>
            <xs:enumeration value="rojo"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

7.3 xs:pattern

Sirve para definir un patrón de caracteres admitidos (en este caso se admite una única letra minúscula de la "a" a la "z"). El valor del patrón tiene que ser una **expresión regular.**

En el siguiente ejemplo se define un elemento llamado “letra” con la restricción de que el único valor admitido es una de las letras minúsculas de la "a" a la "z":

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

> *Si queréis ampliar la información acerca de expresiones regulares os recomiendo que realicéis una breve búsqueda. En lo concierne a este modulo, solo veremos los ejemplos más básicos en algún ejercicio.*

7.4 xs:length

Sirve para especificar una longitud fija. En el siguiente ejemplo se define un elemento llamado “clave” con la restricción de que su valor tiene que ser una cadena de, exactamente, doce caracteres:

```
<xs:element name="clave">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:length value="12"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

8. Elemento vacio

Para definir elementos vacios, es decir, que no contienen atributos, elementos hijos, y tampoco contienen información entres sus etiquetas, existen 4 modos distintos de hacerlo.

1) Definiendolo como elemento complejo y dejándolo vacio

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

2) Declarandolo como un elemento simple con la restricción de que no puede tener información.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="myEmptyElement">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:maxLength value="0"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
</xs:schema>
```

3) Declarandolo como un elemento simple que tiene prefijado un valor “vacio”.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="myEmptyElement" type="xs:string" fixed=""/>
</xs:schema>
```

4) Definimos el elemento pero no especificamos su contenido

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

9. SimpleContent

A lo largo del documento hemos visto la forma de validar distintas etiquetas, pero falta un caso concreto por definir:

```
<employee id="1">John Smith</employee>
```

Se trata de un elemento complejo que contiene un elemento simple. La validación seria la siguiente:

```
<xs:element name="employee">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="xs:ID"/>
            </xs:extension  >
        </xs:simpleContent >
    </xs:complexType >
</xs:element>
```

Dicho de otra forma el elemento **employee** es de tipo complejo que extiende un tipo simple **xs:string** y le añade un atributo.

9.1 Como aplicar restricciones a este tipo de elementos

En este caso particular resulta complicado porque la gramática de XSD no nos permite utilizar **xs:extension** y **xs:restriction** al mismo tiempo. La solución será la siguiente:

```
<xs:simpleType name="employeeNombre">
    <xs:restriction base="xs:string">
        <xs:maxLength value="50"/>
    </xs:restriction>
</xs:simpleType>

<xs:element name="employee">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="employeeNombre">
                <xs:attribute name="id" type="xs:ID"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

Primero deberíamos definir un elemento de tipo simple en el que especificamos que **employeeNombre** es un elemento derivado de **xs:string** con una restricción (longitud máxima de 50 caracteres).

A continuación repetimos el proceso anterior, pero en vez de extender el tipo xs:string, ahora extenderemos el tipo **xs:employeNombre**