EBook Gratis

APRENDIZAJE xsd

Free unaffiliated eBook created from
Stack Overflow contributors.

#xsd

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: xsd

It is an unofficial and free xsd ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official xsd.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con xsd

Observaciones

El esquema XML es un lenguaje y un marco para validar documentos XML.

Un documento XML que está **bien formado** , en el sentido de que es sintácticamente conforme a la especificación XML, puede probarse la **validez de** un esquema. La distinción entre buena formación, que es absoluta, y validez, que es relativa a un esquema, es primordial.

La validación abarca:

- Verificar si el documento XML cumple con requisitos adicionales, tales como los elementos que tienen ciertos nombres, restricciones en el contenido de los elementos, restricciones de coherencia (claves primarias, unicidad, etc.), valores de atributos o texto que coinciden con ciertos tipos.
- Tras el éxito, la conversión de la instancia del modelo de datos de entrada (llamada XML Infoset) a una instancia de salida (PSVI: Post-Schema-Validation Infoset), donde los elementos y atributos se anotan con información de tipo, donde se han completado los valores predeterminados, etc.

El esquema XML se introdujo para abordar los requisitos que la validación de DTD no pudo abordar, entre otros, un sistema de tipos más completo que incluye un amplio conjunto de tipos integrados, restricciones de tipo y capacidades de extensión, y más control sobre la restricción del diseño de elementos.

Versiones

Versión	Fecha de lanzamiento
1.0	2001-05-02
1.0, segunda edición	2004-10-28
1.1	2012-04-05

El Esquema XML 1.0 fue aprobado como una Recomendación del W3C en mayo de 2001, y la segunda edición que incorporó erratas se publicó como una Recomendación del W3C varios años después.

XML Schema 1.1 se convirtió en una recomendación de W3C en 2012, que corrigió más errores y agregó otras mejoras, a la vez que era mayormente compatible con versiones anteriores.

Examples

Instalación o configuración

XSD, Definición de esquema XML, es un lenguaje que describe la estructura de los documentos XML. Los archivos XSD se pueden usar para validar un archivo XML. El proceso de hacer esto dependerá de lo que elija para implementarlo. Se debe tener cuidado para garantizar que el motor de validación que utiliza sea compatible con la versión deseada de XSD.

Lea Empezando con xsd en línea: https://riptutorial.com/es/xsd/topic/2907/empezando-con-xsd

Capítulo 2: xs: complexType

Introducción

A xs: complexType proporciona una descripción del contenido de un elemento XML en el documento de instancia. La definición de xs: complexType se puede hacer globalmente, en cuyo caso tiene un nombre y se puede reutilizar dentro del esquema, o puede estar in situ y solo usarse dentro del contexto que se declara.

Parámetros

Atributos	Descripción
resumen	Cuando se establece en verdadero, el tipo complejo no se puede usar directamente en un documento XML de instancia a través de xsi: type. Sin embargo, se puede utilizar como el tipo base para una definición de elemento. (predeterminado falso): solo válido para el nivel raíz xs: complexType's
bloquear	Limita los tipos que se pueden usar en un documento de instancia XML (el valor predeterminado del atributo xs: schemas blockDefault se establece de manera predeterminada; de lo contrario, el valor predeterminado es vacío, los valores '#all' | una lista de ('extensión', 'lista', 'union') separados por espacios en blanco).
final	Los límites de los tipos derivados del uso de este tipo en ciertas formas dentro del esquema (por defecto, el valor del atributo xs: schemas finalDefault, si está configurado, de lo contrario, el valor predeterminado es vacío, #aleros '| o una lista de (' extensión ',' lista ',' union ') separados por espacios en blanco) - Solo válido para el nivel raíz xs: complexType's
carné de identidad	El id del elemento de esquema (opcional)
mezclado	Indica que el elemento XML de la instancia puede contener contenido mixto (el valor predeterminado es falso)
nombre	El nombre de xs: complexType: solo válido para el nivel raíz xs: complexType's
alguna	Se permite cualquier otro atributo que no esté en el espacio de nombres ' http://www.w3.org/2001/XMLSchema '.
----------	------
Elementos	**Descripción**

Atributos	Descripción
----------	------
xs: anotacion	Proporciona la capacidad de agregar documentación y datos legibles por máquina.
xs: simpleContent	Se utiliza cuando el xs: complexType deriva de un xs: simpleType.
xs: complexContent	Se utiliza cuando el xs: complexType deriva de otro xs: complexType.
xs: grupo	Agrega los elementos de un grupo xs: a la definición xs: complexType
xs: todos	Agrega los elementos de un xs: todos a la definición de xs: complexType
xs: elección	Agrega los elementos de una opción xs: a la definición xs: complexType
xs: secuencia	Agrega los elementos de una secuencia xs: a la definición xs: complexType
xs: atributo	Agrega el atributo xs: a la definición xs: complexType
xs: attributeGroup	Agrega el xs: attributeGroup a la definición de xs: complexType
xs: anyAttribute	Agrega el atributo xs: anyAttribute a la definición xs: complexType

Observaciones

Derivando de un xs: complexType

Cuando un xs: complexType deriva de otro xs: complexType puede hacerlo por *extensión* o *restricción* .

- extensión: el tipo de derivación toma todo lo definido en el tipo base y lo agrega.
- restricción: el tipo de derivación solo toma partes seleccionadas del tipo base, solo permite las partes que desea, no se pueden agregar elementos adicionales.

Derivando de un xs: simpleType

Cuando un xs: complexType deriva de un xs: simpleType puede hacerlo mediante una *extensión* , en cuyo caso puede agregar atributos al tipo resultante, pero no elementos.

Tipo de contenido

Conceptualmente, un xs: complexType contiene contenido *simple* o *complejo* . Si el xs: complexType deriva de un tipo basado en xs: anySimpleType (xs: int, xs: string, etc.), entonces es *simple* . Si se deriva de un xs: complexType que contiene contenido *complejo* , entonces es

complejo (si el xs: complexType no se deriva de un tipo, entonces también es complejo).

Examples

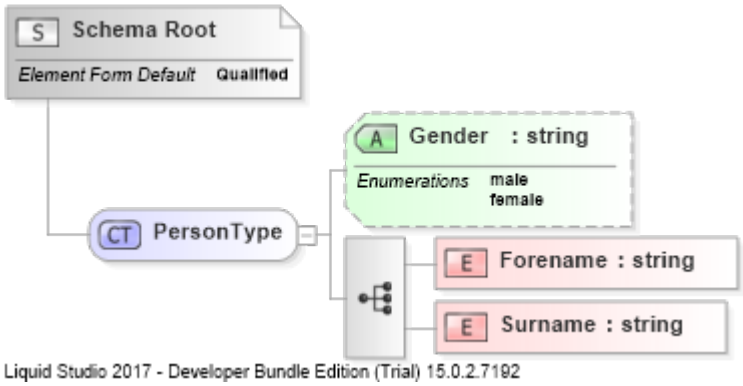
Tipo complejo global con secuencia y atributos

Este ejemplo muestra una definición global simple de un complexType. La definición se considera global ya que es un elemento secundario del esquema xs :. Los tipos definidos globalmente se pueden usar en cualquier otra parte del esquema.

Esta es la forma más común para declarar un xs: complexType global, define los elementos secundarios utilizando xs: sequence, xs: choice o xs: all, y opcionalmente también tiene atributos.

Nota: como se define globalmente, debe tener un nombre único dentro del conjunto de esquemas.

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Created with Liquid Studio 2017 (https://www.liquid-technologies.com)-->
<xs:schema elementFormDefault="qualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:complexType name="PersonType">
        <xs:sequence>
            <xs:element name="Forename" type="xs:string" />
            <xs:element name="Surname" type="xs:string" />
        </xs:sequence>
        <xs:attribute name="Gender">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="male" />
                    <xs:enumeration value="female" />
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
</xs:schema>
```

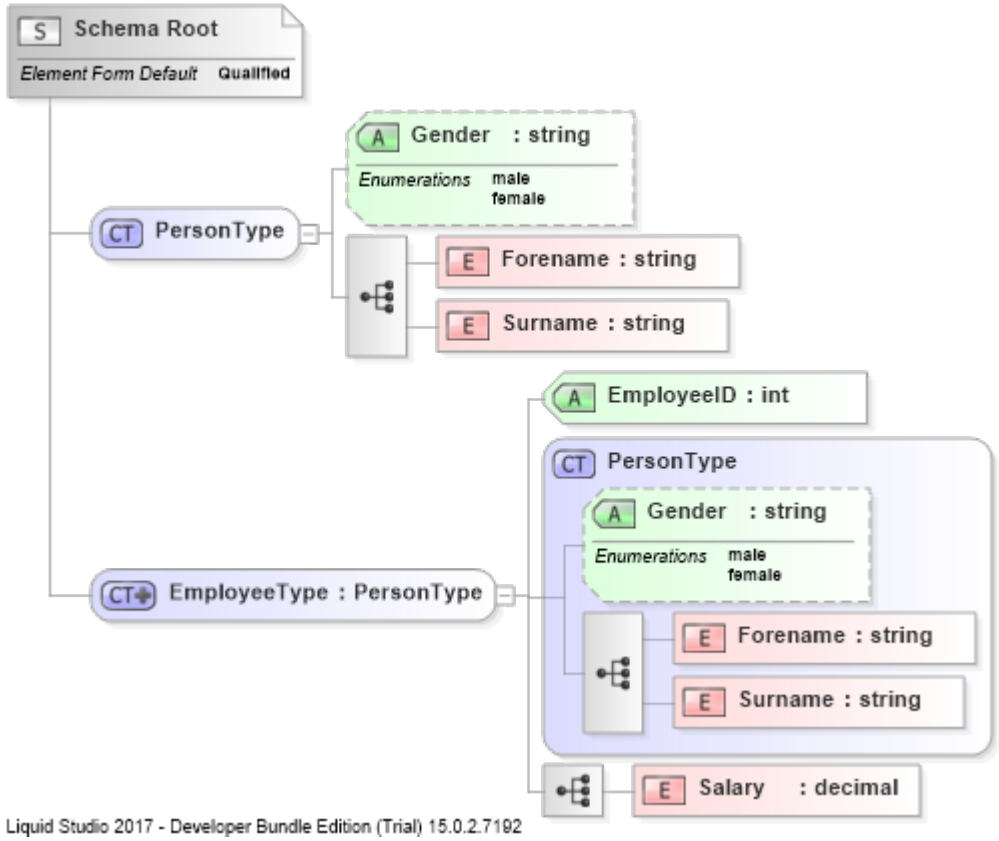


Creando un xs: complexType global extendiendo un xs: complexType existente

En este ejemplo, estamos creando un nuevo xs: complexType (EmployeeType) basado en un xs: complexType existente (PersonType).

La construcción de este es un poco más complicada. Debido a que la base xs: complexType (PersonType) se considera *compleja* (más sobre esto más adelante) agregamos el elemento <xs: complexContent>. Luego, debido a que estamos extendiendo PersonType, agregamos el elemento <xs: extension base = "PersonType">. Dentro de la etiqueta xs: extension podemos agregar un compositor (xs: all / xs: choice / xs: sequence) y cualquier atributo adicional.

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Created with Liquid Studio 2017 (https://www.liquid-technologies.com)-->
<xs:schema elementFormDefault="qualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:complexType name="PersonType">
        <xs:sequence>
            <xs:element name="Forename" type="xs:string" />
            <xs:element name="Surname" type="xs:string" />
        </xs:sequence>
        <xs:attribute name="Gender">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="male" />
                    <xs:enumeration value="female" />
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="EmployeeType">
        <xs:complexContent>
            <xs:extension base="PersonType">
                <xs:sequence>
                    <xs:element name="Salary" type="xs:decimal" />
                </xs:sequence>
                <xs:attribute name="EmployeeID" type="xs:int" use="required" />
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```



Creando un xs: complexType global restringiendo un xs: complexType existente

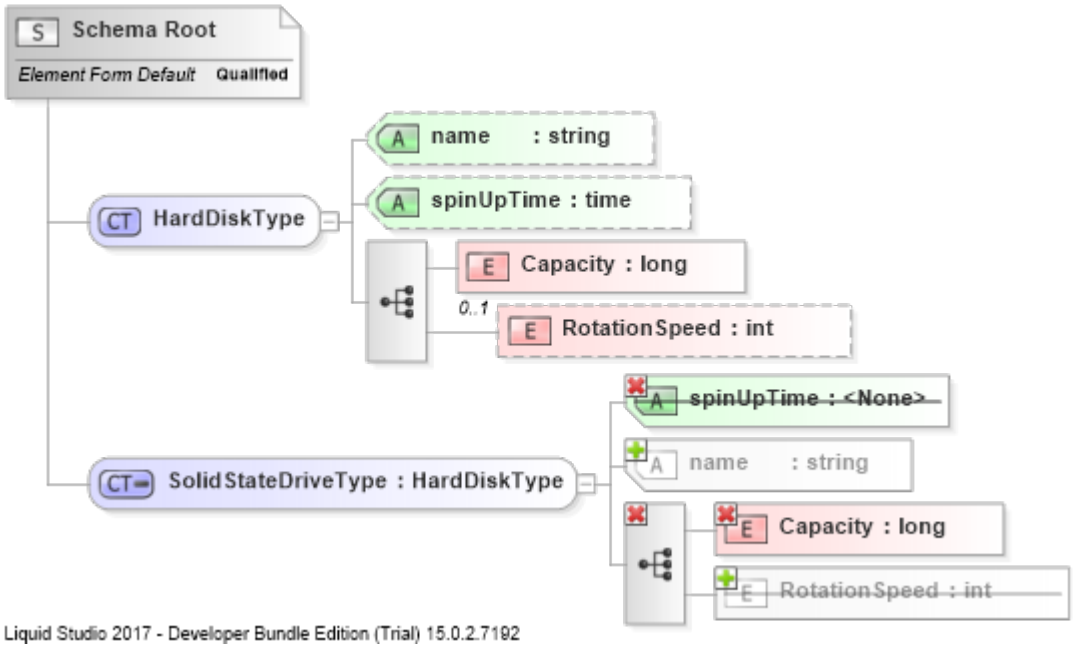
Aquí es donde las cosas se ponen un poco difíciles. Ahora estamos restringiendo un xs: complexType existente. Nuestro SolidStateDriveType deriva de HardDiskType pero elimina el atributo spinUpTime y el elemento RotationSpeed.

Observe que el enfoque para tratar con atributos y elementos es diferente. Para eliminar un atributo, debe volver a declararlo y establecer su *uso* como *prohibido* . Para los elementos que simplemente no volver a declararlos, se excluirán, de hecho, debe volver a declarar cualquier elemento que desee mantener en el nuevo tipo.

Concepto clave para los tipos restringidos *: debe ser posible cargar un elemento de instancia XML resultante del tipo restringido en el tipo base, en otras palabras, el tipo restringido debe poder "ajustarse" al tipo base. Por lo tanto, no puede excluir un atributo o elemento obligatorio, para excluirlo en el tipo restringido debe ser opcional en el tipo base. Si cambia las reglas de tipo / faceta de un elemento o atributo en el tipo restringido, las nuevas reglas de tipo / faceta deben ser compatibles con el tipo base, por lo que si el tipo base era corto, el tipo restringido podría ser un byte, pero no mucho*

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Created with Liquid Studio 2017 - Developer Bundle Edition (Trial) 15.0.2.7192
(https://www.liquid-technologies.com)-->
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:complexType name="HardDiskType">
        <xs:sequence>
            <xs:element name="Capacity" type="xs:long" />
            <xs:element name="RotationSpeed" type="xs:int" minOccurs="0" />
```

```
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" />
        <xs:attribute name="spinUpTime" type="xs:time" />
    </xs:complexType>
    <xs:complexType name="SolidStateDrive">
        <xs:complexContent>
            <xs:restriction base="HardDiskType">
                <xs:sequence>
                    <xs:element name="Capacity" type="xs:long" />
                </xs:sequence>
                <xs:attribute name="spinUpTime" use="prohibited" />
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```



Lea xs: complexType en línea: https://riptutorial.com/es/xsd/topic/9047/xs--complextype

Capítulo 3: xs: esquema

Introducción

Describe los elementos, atributos y tipos que son válidos en un documento de instancia XML. Un esquema XML (XSD) debe contener un solo elemento raíz xs: elemento de esquema.

Parámetros

Atributos	Descripción
attributeFormDefault	Indica si los atributos en el documento de instancia XML tienen que ser calificados con un espacio de nombres (predeterminado no calificado)
bloqueo de defecto	El valor predeterminado del atributo de *bloque* que se aplica a xs: complexType y xs: element. Define las reglas para bloquear la derivación / sustitución en el documento de instancia (vacío predeterminado, es decir, no bloquear nada)
atributos predeterminados	**(XSD 1.1)** Especifica un xs: attributeGroup que se asociará con todos los elementos xs: complexType y xs: dentro del esquema (opcional).
elementFormDefault	Indica si el elemento en el documento de instancia XML debe calificarse con un espacio de nombres (por defecto no calificado). **Nota** : casi sin excepción, todos los esquemas configuran esto como ' *calificado* '.
finalDefault	El valor de atributo *final* predeterminado utilizado en xs: complexType y xs: element. Define las reglas para bloquear la derivación / sustitución en el esquema (vacío predeterminado, es decir, no bloquear nada)
carné de identidad	El id del elemento de esquema (opcional)
targetNamespace	Califica todos los elementos y atributos (y componentes definidos globalmente) definidos dentro de este esquema.
versión	La versión del esquema, esta es la versión del documento, no esta versión XSD (es decir, Soap 1.2, FpML 4.2, etc.)
xpathDefaultNamespace	**(XSD 1.1)** El valor predeterminado para el atributo *xpathDefaultNamespace* que se usa en xs: selector, xs: field, xs: alternative & xs: assert (especifica el espacio de nombres predeterminado que se usará en las expresiones XPath)

Atributos	Descripción
alguna	Se permite cualquier otro atributo que no esté en el espacio de nombres ' http://www.w3.org/2001/XMLSchema '.
Elementos	**Descripción**
xs: anotacion	Proporciona la capacidad de agregar documentación y datos legibles por máquina.
xs: incluir	Se utiliza para incluir un esquema con el mismo targetNamespace o sin targetNamespace (ver esquemas de camaleón).
xs: importar	Se utiliza para incluir un esquema con un targetNamespace diferente al principal.
xs: redefinir	Utilizado para incluir un esquema con el mismo targetNamespace (o sin targetNamespace), y modificar las definiciones xs: simpleType, xs: complexType, xs: group o xs: attributeGroup contenidas en él (aquí hay dragones ...)
xs: simpleType	Define un tipo simple global (con nombre) que luego puede ser referenciado y reutilizado.
xs: complexType	Define un tipo complejo global (con nombre) que luego puede ser referenciado y reutilizado.
xs: grupo	Define un grupo global (con nombre) de elementos que luego pueden ser referenciados y reutilizados.
xs: attributeGroup	Define un grupo global (con nombre) de atributos que luego pueden ser referenciados y reutilizados.
xs: atributo	Define un atributo global (con nombre) que luego puede ser referenciado y reutilizado.
xs: elemento	Define un elemento global (con nombre) que luego puede ser referenciado y reutilizado, o utilizado como base de un documento de instancia XML.
xs: notación	-
xs: defaultOpenContent	**(XSD 1.1)** Especifica reglas para permitir que se permitan elementos adicionales dentro de cada elemento xs: complexType y xs: dentro del esquema.

Examples

Basic xs: esquema

Muestra un esquema muy básico.

Nota: por convención, elementFormDefault está configurado como ' *calificado* ', en el mundo real será difícil encontrar un esquema que no lo establezca (¡así que inclúyalo en sus esquemas!).

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Created with Liquid Studio 2017 - (https://www.liquid-technologies.com)-->
<xs:schema elementFormDefault="qualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Person">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Forename" type="xs:string" />
                <xs:element name="Surname" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

xs: atributo elementFormDefault del esquema

Por convención, elementFormDefault siempre se establece en *calificado* , pero veamos lo que realmente hace.

Primero con elementFormDefault establecido en calificado.

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Created with Liquid Studio 2017 (https://www.liquid-technologies.com)-->
<xs:schema elementFormDefault="qualified"
           targetNamespace="http://base.com"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="MyBaseElement">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="ChildA" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Documento XML de muestra

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid Studio 2017 (https://www.liquid-technologies.com) -->
<b:MyBaseElement xmlns:b="http://base.com"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://base.com ElementFormDefault_qualified.xsd">
    <b:ChildA>string</b:ChildA>
</b:MyBaseElement>
```

Observe que el elemento ChildA también se debe calificar con el espacio de nombres 'b'.

Ahora veamos con elementFormDefault configurado como no calificado.

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Created with Liquid Studio 2017 (https://www.liquid-technologies.com)-->
<xs:schema elementFormDefault="unqualified"
           targetNamespace="http://base.com"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="MyBaseElement">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="ChildA" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Documento XML de muestra

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid Studio 2017 (https://www.liquid-technologies.com) -->
<b:MyBaseElement xmlns:b="http://base.com"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://base.com ElementFormDefault_unqualified.xsd">
    <ChildA>string</ChildA>
</b:MyBaseElement>
```

Observe esta vez que solo el elemento definido globalmente MyBaseElement está calificado con el espacio de nombres 'b', el elemento interno ChildA (que está definido en el lugar dentro del esquema) no está calificado.

En el último ejemplo, vimos que los elementos definidos globalmente deben calificarse en el documento de instancia XML, pero los elementos definidos in situ no. Pero esto no solo significa el elemento raíz, si tiene elementos definidos globalmente a los que se hace referencia, también es necesario calificarlos.

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Created with Liquid Studio 2017 (https://www.liquid-technologies.com)-->
<xs:schema elementFormDefault="unqualified"
           targetNamespace="http://base.com"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="MyBaseElement">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="ChildA" type="xs:string" />
                <xs:element xmlns:q1="http://base.com" ref="q1:MyElement" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="MyElement" type="xs:string" />
</xs:schema>
```

Documento XML de muestra

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<!-- Created with Liquid Studio 2017 (https://www.liquid-technologies.com) -->
<b:MyBaseElement xmlns:b="http://base.com"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://base.com ElementFormDefault_unqualified.xsd">
    <ChildA>string</ChildA>
    <b:MyElement>string</b:MyElement>
</b:MyBaseElement>
```

Tenga en cuenta que MyElement también necesita calificar como se define globalmente.

En conclusión, si tiene elementFormDefault configurado en calificado, entonces todo debe ser calificado con un espacio de nombres (ya sea a través de un alias de espacio de nombres o configurando el nombre de archivo predeterminado xmlns = "..."). Sin embargo, elementFormDefault se establece en no calificado, las cosas se complican y debe realizar un examen bastante profundo de los esquemas para determinar si las cosas deben ser calificadas o no.

¡Supongo que es por eso que elementFormDefault siempre está configurado como calificado!

Lea xs: esquema en línea: https://riptutorial.com/es/xsd/topic/9052/xs--esquema

Creditos

S. No	Capítulos	Contributors
1	Empezando con xsd	Beth Whitezel, Community, Ghislain Fourny, whrrgarbl, Wolfgang Schindler
2	xs: complexType	Sprotty
3	xs: esquema	Sprotty