Kostenloses eBook

LERNEN

xsd

Free unaffiliated eBook created from
Stack Overflow contributors.

#xsd

Inhaltsverzeichnis

Über

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

Kapitel 1: Erste Schritte mit xsd

Bemerkungen

XML Schema ist eine Sprache und ein Framework zur Validierung von XML-Dokumenten.

Ein XML-Dokument, **das** in dem Sinn, dass es syntaktisch mit der XML-Spezifikation übereinstimmt, **wohlgeformt** ist, kann auf **Gültigkeit** anhand eines Schemas getestet werden. Die Unterscheidung zwischen Wohlgeformtheit, die absolut ist, und Gültigkeit, die relativ zu einem Schema ist, ist von größter Bedeutung.

Validierung umfasst:

- Überprüfung, ob das XML-Dokument zusätzliche Anforderungen erfüllt, z. B. Elemente mit bestimmten Namen, Einschränkungen für den Inhalt von Elementen, Konsistenzbedingungen (Primärschlüssel, Eindeutigkeit usw.), Attributwerte oder Text, der bestimmten Typen entspricht.
- Bei Erfolg Konvertierung der Eingabedatenmodellinstanz (XML-Infoset) in eine Ausgabeinstanz (PSVI: Infoset für Post-Schema-Validation), in der Elemente und Attribute mit Typinformationen versehen werden, in denen Standardwerte usw. eingetragen sind.

XML Schema wurde eingeführt, um Anforderungen zu erfüllen, die bei der DTD-Validierung nicht berücksichtigt wurden, unter anderem ein umfassenderes Typsystem, das eine Vielzahl von integrierten Typen, Typenbeschränkungen und Erweiterungsfunktionen sowie mehr Kontrolle über die Einschränkung des Elementlayouts umfasst.

Versionen

Ausführung	Veröffentlichungsdatum
1,0	2001-05-02
1.0, zweite Auflage	2004-10-28
1.1	2012-04-05

XML Schema 1.0 wurde im Mai 2001 als W3C-Empfehlung genehmigt, und die zweite Ausgabe mit Errata wurde einige Jahre später als W3C-Empfehlung veröffentlicht.

XML Schema 1.1 wurde 2012 zu einer W3C-Empfehlung, die weitere Fehler behebte und weitere Verbesserungen hinzufügte, während es mit früheren Versionen weitgehend kompatibel war.

Examples

Installation oder Setup

XSD (XML Schema Definition) ist eine Sprache, die die Struktur von XML-Dokumenten beschreibt. XSD-Dateien können verwendet werden, um eine XML-Datei zu überprüfen. Der Vorgang hängt davon ab, mit was Sie ihn implementieren möchten. Es ist darauf zu achten, dass die von Ihnen verwendete Validierungs-Engine mit der gewünschten Version von XSD kompatibel ist.

Erste Schritte mit xsd online lesen: https://riptutorial.com/de/xsd/topic/2907/erste-schritte-mit-xsd

Kapitel 2: xs: complexType

Einführung

Ein xs: complexType enthält eine Beschreibung des Inhalts eines XML-Elements im Instanzdokument. Die Definition von xs: complexType kann global vorgenommen werden. In diesem Fall hat es einen Namen und kann innerhalb des Schemas erneut verwendet werden, oder es kann inplace sein und nur innerhalb des deklarierten Kontexts verwendet werden.

Parameter

Attribute	Beschreibung
abstrakt	Bei true kann der komplexe Typ nicht direkt in einem XML-Instanzdokument über xsi: type verwendet werden. Es kann jedoch als Basistyp für eine Elementdefinition verwendet werden. (Standardwert false) - Nur gültig für Stammebene xs: complexType
Block	Beschränkt die Typen, die in einem XML-Instanzdokument verwendet werden können (Standardwert ist der Wert des Attributs xs: schemas blockDefault, falls festgelegt, andernfalls leer, Werte '#all' | eine Liste von ('Erweiterung', 'Liste', 'union') getrennt durch Leerzeichen).
Finale	Begrenzt das Ableiten von Typen von der Verwendung dieses Typs auf bestimmte Arten innerhalb des Schemas (standardmäßig wird der Wert des Attributs "finalDefault" xs: schemas verwendet, sofern festgelegt, andernfalls leer, Werte "#all" | oder eine Liste der ("Erweiterung", "- Liste") ',' union ') getrennt durch Leerzeichen) - Nur gültig für Stammebene xs: complexType's
Ich würde	Die ID des Schemaelements (optional)
gemischt	Gibt an, dass das Instanz-XML-Element gemischten Inhalt enthalten kann (Standardeinstellung ist "false").
Name	Der Name von xs: complexType - Nur gültig für die Stammebene xs: complexType
irgendein	Alle anderen Attribute, die nicht im Namensraum ' http://www.w3.org/2001/XMLSchema ' liegen, sind zulässig.
----------	------
Elemente	**Beschreibung**
----------	------

Attribute	Beschreibung
xs: Anmerkung	Bietet die Möglichkeit, Dokumentation und maschinenlesbare Daten hinzuzufügen.
xs: simpleContent	Wird verwendet, wenn xs: complexType von einem xs: simpleType abgeleitet ist.
xs: complexContent	Wird verwendet, wenn xs: complexType von einem anderen xs: complexType stammt.
xs: Gruppe	Fügt der xs: complexType-Definition die Elemente aus einer xs: group hinzu
xs: alle	Fügt der xs: complexType-Definition die Elemente aus einer xs: all hinzu
xs: Wahl	Fügt der xs: complexType-Definition die Elemente aus einer xs: choice hinzu
xs: Sequenz	Fügt der xs: complexType-Definition die Elemente aus einer xs: -Sequenz hinzu
xs: Attribut	Fügt der xs: complexType-Definition das Attribut xs: hinzu
xs: attributeGroup	Fügt die xs: attributeGroup der xs: complexType-Definition hinzu
xs: anyAttribute	Fügt der xs: complexType-Definition das xs: anyAttribute hinzu

Bemerkungen

Ableitung von einem xs: complexType

Wenn ein xs: complexType von einem anderen xs: complexType abgeleitet ist, kann dies über eine *Erweiterung* oder *Einschränkung erfolgen* .

- Erweiterung - Der abgeleitete Typ übernimmt alles, was im Basistyp definiert ist, und fügt ihn hinzu.
- Einschränkung - Der Ableitungstyp nimmt nur ausgewählte Teile vom Basistyp auf, wobei nur die gewünschten Teile zugelassen werden. Es können keine weiteren Elemente hinzugefügt werden.

Ableitung von einem xs: simpleType

Wenn ein xs: complexType von einem xs: simpleType abgeleitet ist, kann dies über die *Erweiterung erfolgen* . In diesem Fall kann er dem resultierenden Typ Attribute hinzufügen, jedoch keine Elemente.

Inhaltstyp

Konzeptionell enthält ein xs: complexType entweder *einfachen* oder *komplexen* Inhalt. Wenn xs: complexType von einem Typ abgeleitet wird, der auf xs: anySimpleType (xs: int, xs: string usw.) basiert, ist dies *einfach* . Wenn es von einem xs: complexType abgeleitet ist, der *komplexen* Inhalt enthält, ist es selbst *komplex* (wenn der xs: complexType nicht von einem Typ stammt, ist er auch komplex).

Examples

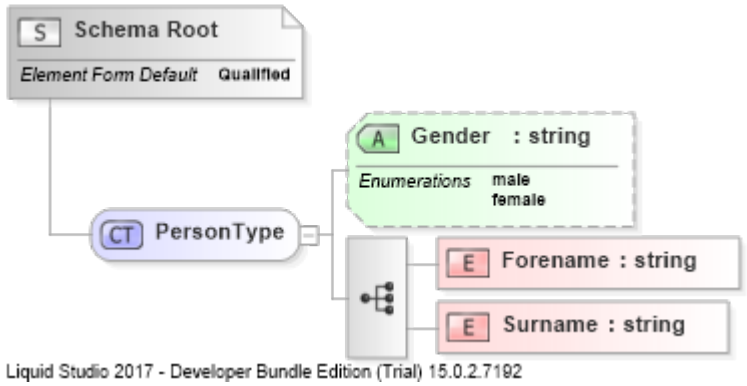
Globaler ComplexType mit Sequenz und Attributen

Dieses Beispiel zeigt eine einfache globale Definition eines complexType. Die Definition wird als global betrachtet, da sie dem xs: Schema untergeordnet ist. Global definierte Typen können an anderer Stelle im Schema verwendet werden.

Dies ist die gebräuchlichste Form zum Deklarieren eines globalen xs: complexType. Sie definiert die untergeordneten Elemente mit einer xs: sequence, xs: choice oder xs: all und hat optional auch Attribute.

Hinweis: Da es sich um ein global definiertes Element handelt, muss es innerhalb des Schemasets einen eindeutigen Namen haben.

```
<?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>
```

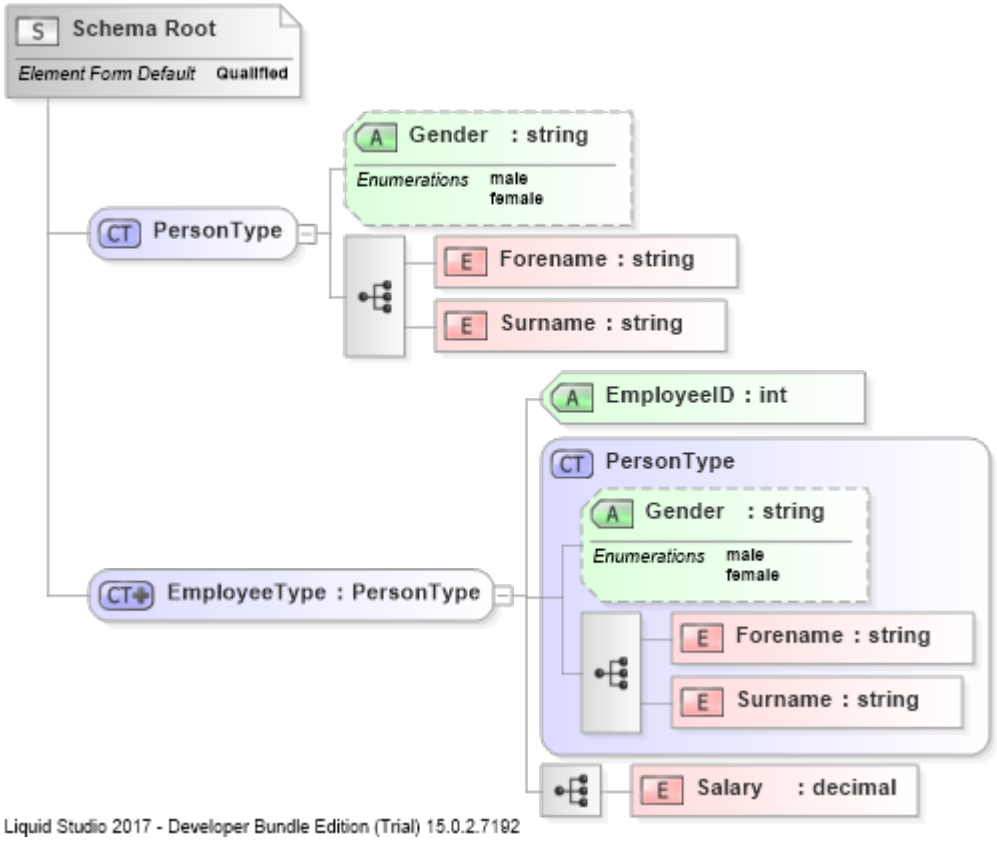


Erstellen eines globalen xs: complexType durch Erweitern eines vorhandenen xs: complexType

In diesem Beispiel erstellen wir einen neuen xs: complexType (EmployeeType), der auf einem vorhandenen xs: complexType (PersonType) basiert.

Der Aufbau ist etwas komplizierter. Da die Basis xs: complexType (PersonType) als *komplex betrachtet wird* (mehr dazu unten), fügen wir das Element <xs: complexContent> hinzu. Dann fügen wir das Element <xs: extension base = "PersonType"> hinzu, da wir PersonType erweitern. Innerhalb des xs: extension-Tags können wir einen Compositor (xs: all / xs: choice / xs: sequence) und zusätzliche Attribute hinzufügen.

```
<?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>
```



Erstellen eines globalen xs: complexType durch Einschränken eines vorhandenen xs: complexType

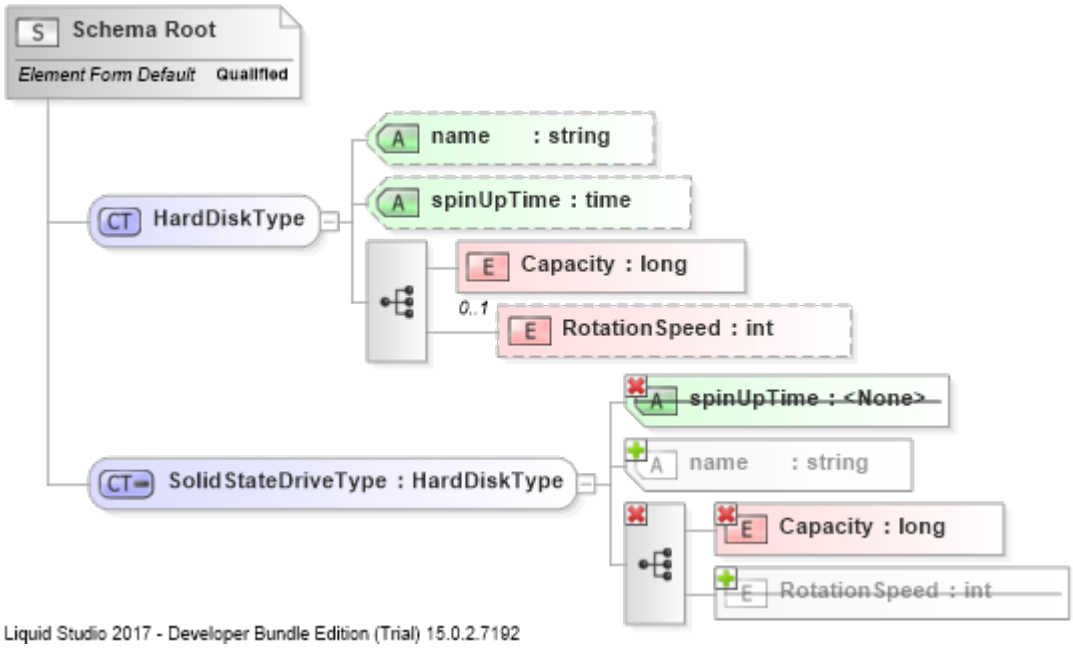
Hier wird es etwas knifflig. Wir beschränken jetzt einen vorhandenen xs: complexType. Unser SolidStateDriveType wird von HardDiskType abgeleitet, entfernt jedoch das Attribut spinUpTime und das RotationSpeed-Element.

Beachten Sie, dass der Ansatz für den Umgang mit Attributen und Elementen unterschiedlich ist. Um ein Attribut zu entfernen, müssen Sie es erneut deklarieren und seine *Verwendung* auf " *gesperrt" setzen* . Wenn Elemente einfach nicht erneut deklariert werden, werden sie ausgeschlossen. In der Tat müssen Sie alle Elemente, die Sie im neuen Typ behalten möchten, erneut deklarieren.

Schlüsselkonzept für eingeschränkte Typen *: Es muss möglich sein, ein aus dem eingeschränkten Typ resultierendes XML-Instanzelement in den Basistyp zu laden. Anders ausgedrückt, der eingeschränkte Typ muss in den Basistyp passen. Sie können also ein obligatorisches Attribut oder Element nicht ausschließen, um es im eingeschränkten Typ auszuschließen, muss es im Basistyp optional sein. Wenn Sie die Typ- / Facettenregeln eines Elements oder Attributs im eingeschränkten Typ ändern, müssen die neuen Typ- / Facettenregeln mit dem Basistyp kompatibel sein. Wenn der Basistyp ein kurzer Typ ist, könnte der eingeschränkte Typ jedoch ein Byte sein nicht lange*

```
<?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>
```



xs: complexType online lesen: https://riptutorial.com/de/xsd/topic/9047/xs--complextype

Kapitel 3: xs: schema

Einführung

Beschreibt Elemente, Attribute und Typen, die in einem XML-Instanzdokument gültig sind. Ein XML-Schema (XSD) muss ein einzelnes xs: schema-Element auf Stammebene enthalten.

Parameter

Attribute	Beschreibung
attributeFormDefault	Gibt an, ob Attribute im XML-Instanzdokument mit einem Namespace qualifiziert werden müssen (Standardwert nicht qualifiziert).
blockDefault	Der Standardwert des *Blocks* Attribut, das angewendet wird , xs: complex und xs: element. Definiert die Regeln für das Blockieren der Ableitung / Substitution im Instanzdokument (standardmäßig leer, dh nichts blockieren).
defaultAttributes	**(XSD 1.1)** Gibt eine xs: attributeGroup an, die allen xs: complexType- und xs: -Elementen innerhalb des Schemas zugeordnet wird (optional).
elementFormDefault	Gibt an, ob ein Element im XML-Instanzdokument mit einem Namespace qualifiziert werden muss (Standardwert nicht qualifiziert). **Hinweis** : Fast ausnahmslos setzen alle Schemas auf ' *qualifiziert* '.
finalDefault	Der *endgültige* Standardattributwert, der in xs: complexType und dem xs: -Element verwendet wird. Definiert die Regeln für das Blockieren der Ableitung / Substitution im Schema (standardmäßig leer, dh nichts blockieren).
Ich würde	Die ID des Schemaelements (optional)
targetNamespace	Qualifiziert alle Elemente und Attribute (und global definierten Komponenten), die in diesem Schema definiert sind.
Ausführung	Die Version des Schemas, dies ist die Version des Dokuments, nicht diese XSD-Version (dh Soap 1.2, FpML 4.2 usw.)
xpathDefaultNamespace	**(XSD 1.1)** Der Standardwert für das Attribut *xpathDefaultNamespace,* das in xs: selector, xs: field, xs: alternative und xs: assert verwendet wird (gibt den Standardnamespace an, der in XPath-Ausdrücken verwendet

Attribute	Beschreibung
	wird.)
irgendein	Alle anderen Attribute, die nicht im Namensraum ' http://www.w3.org/2001/XMLSchema ' liegen, sind zulässig.
Elemente	**Beschreibung**
xs: Anmerkung	Bietet die Möglichkeit, Dokumentation und maschinenlesbare Daten hinzuzufügen.
xs: include	Wird verwendet, um ein Schema mit demselben targetNamespace oder keinem targetNamespace (siehe Chamäleon-Schemas) einzuschließen.
xs: import	Wird verwendet, um ein Schema mit einem anderen Ziel-Namespace als dem übergeordneten Element aufzunehmen.
xs: neu definieren	Wird verwendet, um ein Schema mit demselben targetNamespace (oder keinem targetNamespace) einzuschließen und xs: simpleType, xs: complexType, xs: group oder die darin enthaltenen xs: attributeGroup-Definitionen zu ändern (hier Dragons ....)
xs: simpleType	Definiert einen globalen (benannten) einfachen Typ, der dann referenziert und erneut verwendet werden kann.
xs: complexType	Definiert einen globalen (benannten) komplexen Typ, der dann referenziert und erneut verwendet werden kann.
xs: Gruppe	Definiert eine globale (benannte) Gruppe von Elementen, auf die dann verwiesen und erneut verwendet werden kann.
xs: attributeGroup	Definiert eine globale (benannte) Attributgruppe, auf die dann verwiesen und erneut verwendet werden kann.
xs: Attribut	Definiert ein globales (benanntes) Attribut, das dann referenziert und erneut verwendet werden kann.
xs: Element	Definiert ein globales (benanntes) Element, das dann referenziert und erneut verwendet werden kann oder als Basis für ein XML-Instanzdokument verwendet werden kann.
xs: Notation	-
xs: defaultOpenContent	**(XSD 1.1)** Gibt Regeln an, nach denen zusätzliche Elemente in jedem xs: complexType- und xs: -Element innerhalb des Schemas zulässig sind.

Examples

Grundlegendes xs: Schema

Zeigt ein sehr grundlegendes Schema.

Hinweis: Gemäß der Konvention ist elementFormDefault auf ' *qualifiziert* ' gesetzt. In der Realität wird es Ihnen schwer fallen, ein Schema zu finden, in dem dies nicht festgelegt ist.

```
<?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: schema elementFormDefault-Attribut

Per Konvention wird elementFormDefault immer auf *qualifiziert gesetzt* , aber sehen wir uns an, was es tatsächlich tut.

Zuerst mit elementFormDefault auf Qualifiziert.

```
<?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>
```

XML-Beispieldokument

```
<?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>
```

Beachten Sie, dass das Element ChildA ebenfalls mit dem Namespace 'b' qualifiziert werden muss.

Jetzt betrachten wir es mit elementFormDefault, das auf unqualifiziert gesetzt ist.

```
<?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>
```

XML-Beispieldokument

```
<?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>
```

Beachten Sie diesmal, dass nur das global definierte Element MyBaseElement mit dem Namespace 'b' qualifiziert ist. Das innere Element ChildA (das innerhalb des Schemas definiert ist) ist nicht qualifiziert.

Im letzten Beispiel haben wir gesehen, dass die global definierten Elemente im XML-Instanzdokument qualifiziert werden müssen, inplace definierte Elemente jedoch nicht. Dies bedeutet jedoch nicht nur das Wurzelelement. Wenn Sie global definierte Elemente haben, auf die verwiesen wird, müssen diese ebenfalls qualifiziert werden.

```
<?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>
```

XML-Beispieldokument

```
<?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>
```

Beachten Sie, dass MyElement auch qualifiziert werden muss, da es global definiert ist.

Fazit: Wenn elementFormDefault auf qualifiziert gesetzt ist, muss alles mit einem Namespace qualifiziert werden (entweder über einen Namespace-Alias oder durch Festlegen der Standardnamenapce xmlns = "..."). Allerdings ist elementFormDefault auf unqualifiziert gesetzt, die Dinge werden kompliziert und Sie müssen einige eingehende Untersuchungen der Schemata durchführen, um herauszufinden, ob die Dinge qualifiziert werden sollen oder nicht.

Ich gehe davon aus, dass elementFormDefault deshalb immer auf qualifiziert gesetzt ist!

xs: schema online lesen: https://riptutorial.com/de/xsd/topic/9052/xs--schema

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit xsd	Beth Whitezel, Community, Ghislain Fourny, whrrgarbl, Wolfgang Schindler
2	xs: complexType	Sprotty
3	xs: schema	Sprotty