XSD and XML

Table of Contents

[XML 2](#_Toc434776373)

[Short rules 2](#_Toc434776374)

[Namespaces 2](#_Toc434776375)

[XSD 3](#_Toc434776376)

[Referencing a Schema in an XML Document 4](#_Toc434776377)

[Schema jDeveloper examples. 5](#_Toc434776378)

[Schema reuse 7](#_Toc434776379)

# XML

Basic definitions:

* XML stands for E**X**tensible **M**arkup **L**anguage.
* XML was designed to store and transport data.
* XML was designed to be both human- and machine-readable.

XML and HTML were designed with different goals:

* XML was designed to carry data - with focus on what data is
* HTML was designed to display data - with focus on how data looks
* XML tags are not predefined like HTML tags are

Html is structured xml. Here is one example of XML.

<book category="cooking">  
    <title lang="en">Everyday Italian</title>  
    <author>Giada De Laurentiis</author>  
    <year>2005</year>  
    <price>30.00</price>  
  </book>

## Short rules

* Every element begins with <name> and ends with </name> or can be presented like <name/> if there is no value in element. Elements can also contain properties like in previous example lang=”en”.
* In XML, it is illegal to omit the closing tag. All elements **must** have a closing tag.
* XML tags are case sensitive. The tag <Letter> is different from the tag <letter>.
* In XML, all elements **must** be properly nested within each other.
* In XML, the attribute values must always be quoted.
* Special character which should be escaped <(&lt;), >(&gt;), &(&amp;), ' (&apos;), " (&quot;)
* <?xml version="1.0" encoding="UTF-8**"**?> The XML prolog is optional. If it exists, it must come first in the document. IT should be specified, otherwise default is UTF-8.

Html is structured xml. Here is one example of XML.

## Namespaces

To use namespace prefixes in your xml, namespaces for prefix must be defined. For this purposes is used xmlns and some basic rules are:

* The namespace can be defined by an **xmlns** attribute in the start tag of an element.
* The namespace declaration has the following syntax. xmlns:*prefix*="*URI*".
* When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace.
* Namespaces can also be declared the XML root element.
* To not use prefixes you can define default namespace xmlns="*namespaceURI*"

# XSD

XSD is schema language which is used to define rules for creation XML documents.

* The XML Schema language is also referred to as XML Schema Definition (XSD).
* XML Schema is an XML-based alternative to DTD.
* XML Schemas are much more powerful than DTDs.

Main usage of XSD xchema is to define structure of message used in communication. It is important, to be sure, that sender and recipient of message can understand syntax and semantic of exchanged content. One of the greatest strength of XML Schemas is the support for data types.

With support for data types:

* It is easier to describe allowable document content
* It is easier to validate the correctness of data
* It is easier to work with data from a database
* It is easier to define data facets (restrictions on data)
* It is easier to define data patterns (data formats)
* It is easier to convert data between different data type

An XML Schema describes the structure of an XML document. ML Schemas are extensible, because they are written in XML.

With an extensible Schema definition you can:

* Reuse your Schema in other Schemas
* Create your own data types derived from the standard types
* Reference multiple schemas in the same document

Some basic XML rules:

A well-formed XML document is a document that conforms to the XML syntax rules, like:

* it must begin with the XML declaration
* it must have one unique root element
* start-tags must have matching end-tags
* elements are case sensitive
* all elements must be closed
* all elements must be properly nested
* all attribute values must be quoted
* entities must be used for special characters

Important fragments of xsd:

**xmlns:xs=”http://www.w3.org/2001/XMLSchema”** ,indicates that the elements and data types used in the schema come from the http://www.w3.org/2001/XMLSchema” namespace. It also specifies that the elements and data types that come from the “http://www.w3.org/2001/XMLSchema” namespace should be prefixed with **xs:.**

**targetNamespace=”**[**http://www.w3schools.com**](http://www.w3schools.com)**”** indicates that the elements defined by this schema (particularly mentioned in schema) come from the “http://www.w3schools.com” namespace.

**(without :prefix) xmlns=**[**http://www.w3schools.com**](http://www.w3schools.com) indicates that the default namespace is “http://www.w3schools.com”.

**elementFormDefault=”qualified”** , indicates that any elements used by the XML instance document which were declared in this schema must be namespace qualified.

# Referencing a Schema in an XML Document

This XML document has a reference to an XML Schema:

<?xml version=”1.0”?>  
  
<note xmlns=”http://www.w3schools.com”  
xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”  
xsi:schemaLocation=”http://www.w3schools.com note.xsd”>  
  
<to>Tove</to>  
<from>Jani</from>  
<heading>Reminder</heading>  
<body>Don’t forget me this weekend!</body>  
</note>

T**o s**pecify the default namespace declaration, xmlns is used (**xmlns=”http://www.w3schools.com”**). This declaration tells the schema-validator that all the elements used in this XML document are declared in the "http://www.w3schools.com" namespace.

Once you have the XML Schema Instance namespace available, xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance", you can use the schemaLocation attribute. This attribute has two values, separated by a space. The first value is the namespace to use. The second value is the location of the XML schema to use for that namespace, xsi:schemaLocation="http://www.w3schools.com note.xsd" .

## Schema jDeveloper examples.

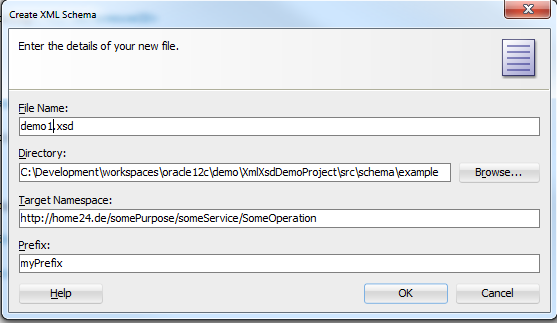
Here we will explain creation schema as well as some core elemnts used to create one, trough pictures and descriptions.

Xsd files are usually stored separately from xml files which they use the. However file organization and structure of folders can be organized as you wish. Usually files are referenced with relative paths where “../” level of parent folder and “./” means current folder.

To create xsd schema in jdeveloper you can do:

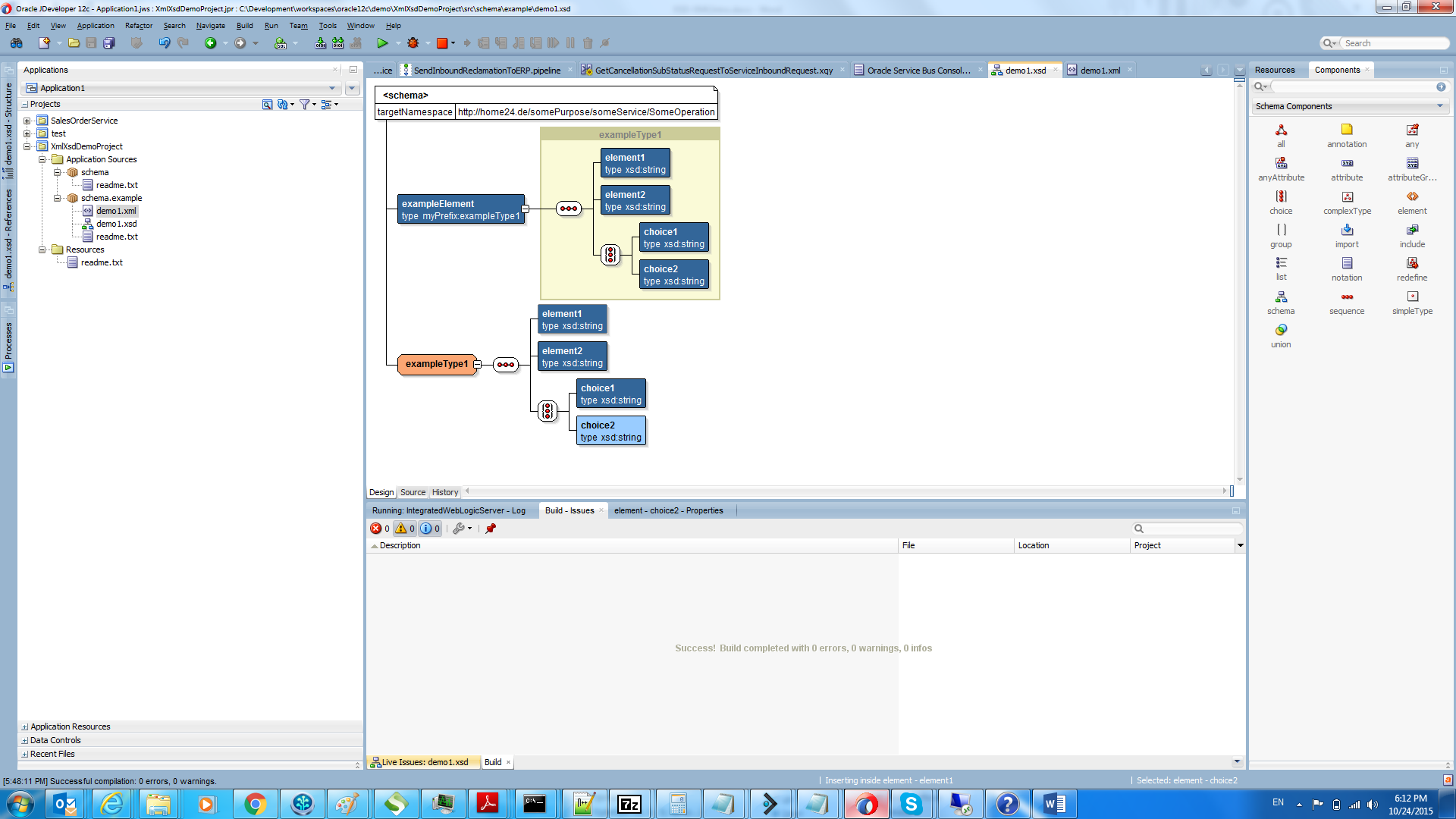
* Right click on any project or folder in project -> Select New->From gallery.
* Chose XML schema option.

Here you will have to define some important values, amongst others, used inside schema. Target namespace is one of them.



Target namespace indicates that the elements defined by this schema come from the mentioned namespace. It is important because same named element can have different meaning. For example “person” used in hospital system can have different attributes and elements in some hospital system and in school system.

Prefix tells that all elements from this namespace will be prefixed with defined prefix. This is also visible from following picture.



Here in same document we can see definition of type exampleType which is defined in xsd where target namespace is defined. Example element is defined as type myprefix:exampleType1. Myprefix is used to prevent confusion while defining elements with types with same names. Namespace make difference and provide instructions to make difference between same named types from different namespace.

Interesting details of source of this schema is below:

**<?xml version="1.0" encoding="windows-1252" ?>**

**<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"**

**xmlns:myPrefix="http://home24.de/somePurpose/someService/SomeOperation"**

**targetNamespace="http://home24.de/somePurpose/someService/SomeOperation" elementFormDefault="qualified">**

**…**

<xsd:element name="exampleElement" type="**myPrefix**:**exampleType1**">

<xsd:annotation>

<xsd:documentation>A sample element</xsd:documentation>

</xsd:annotation>

</xsd:element>

<**xsd**:**complexType** name="**exampleType1**">

<**xsd**:**sequence**>

<**xsd:element name="element1" type="xsd:string**"/>

<xsd:element name="element2" type="xsd:string"/>

<**xsd:choice**>

<xsd:element name="choice1" type="xsd:string"/>

<xsd:element name="choice2" type="xsd:string"/>

</xsd:choice>

</xsd:sequence>

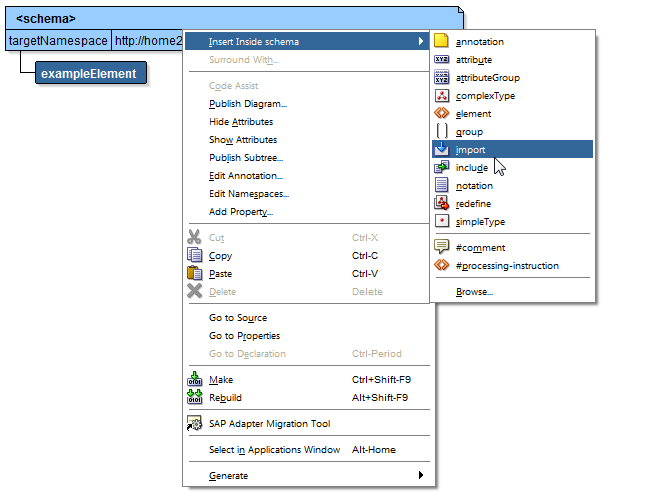
</xsd:complexType>

**…**

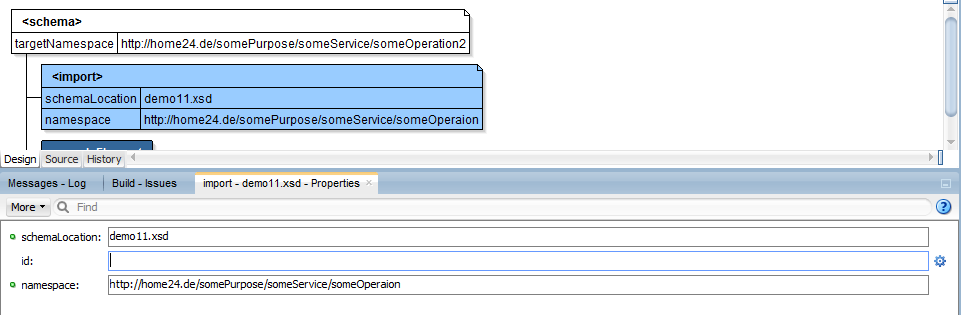
</xsd:schema>

# Schema reuse

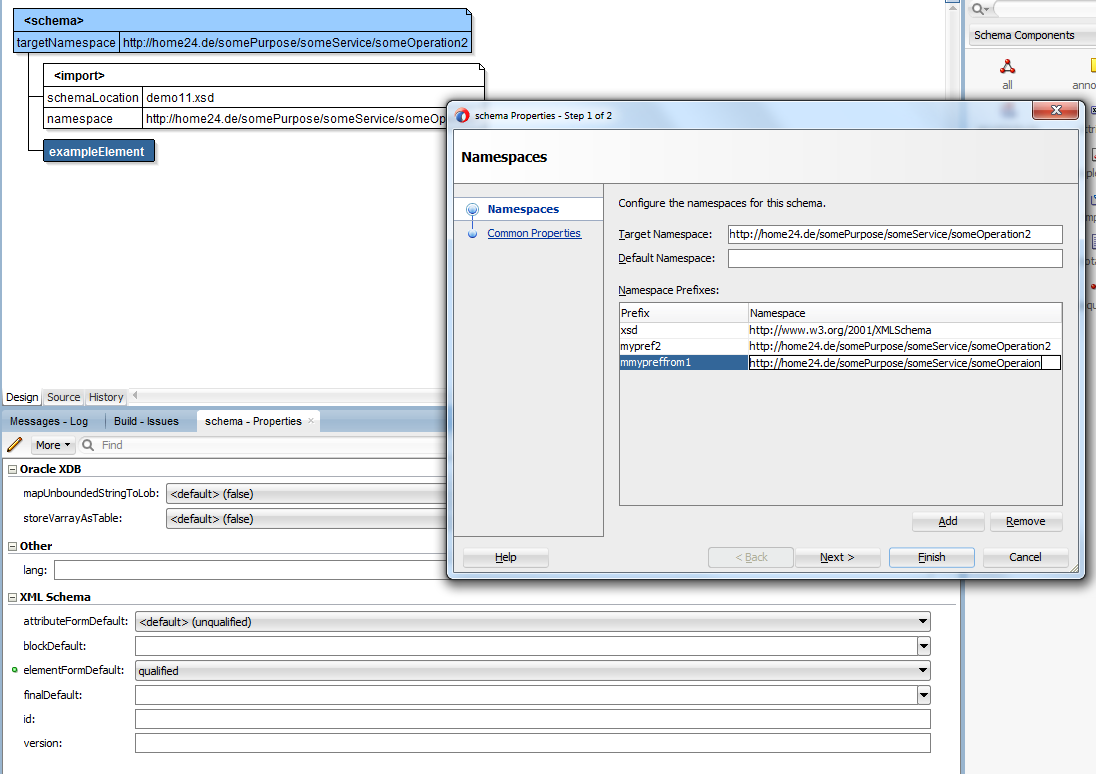
Reuse of schema and types is common way to share definitions between scopes of usage. To import schema you should do right click -> Insert inside schema - > import



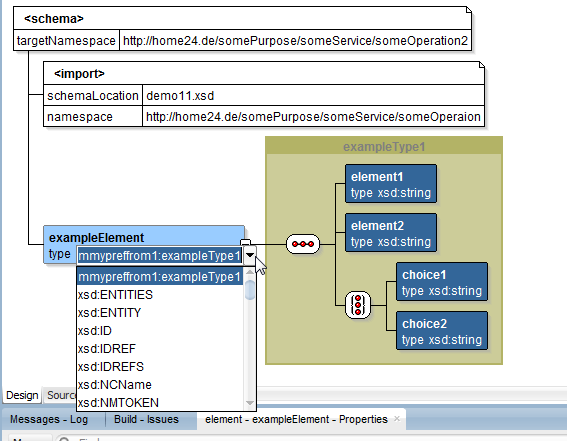
In xsd properties chose file from which you want to import schema (usually relative paths) and namesace.



To be able to use schema we should click on Schema target namespace element and in schema properties, click pen button next to “More” and after that configure (register) namespace which we want to use with prefix.



After this we can use types defined in one schema to create new, more complex types in another.



Note that all of this can be done through source, which is common way.

For more details about xsd components (by using jDevloper) check [Understanding the XSD Component Display in the XSD Visual Editor](http://docs.oracle.com/cd/E24382_01/user.1112/e17455/dev_apps_xml.htm#OJDUG1195)

XML creation

Below is originally generated xml file by jdeveloper.

<?xml version="1.0" encoding="UTF-8" ?>

<exampleElement xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

**xsi:schemaLocation="http://home24.de/somePurpose/someService/someOperaion ../Schemas/demo1/demo11.xsd"**

xmlns="http://home24.de/somePurpose/someService/someOperaion">

<element1></element1>

<element2></element2>

<choice1></choice1>

**<!--**

**<choice2>**

**</choice2>-->**

</exampleElement>

References:

<http://docs.oracle.com/cd/E24382_01/user.1112/e17455/dev_apps_xml.htm#OJDUG4764>

<http://www.w3schools.com/schema/default.asp>