# XSD

L'XML Schema Definition (XSD) è una specifica della struttura dei documenti XML.

**Un XDS consente**

-di definire elementi e attributi,
-di dichiarare il tipo di ciascun elemento e attributo sulla base di un insieme di tipi predefiniti,
-di costruire nuovi tipi con l'aiuto di costruttori di tipi complessi.

Un documento XML che rispetta la descrizione del proprio XSD, si dice validato rispetto ad esso.

**L'XSD è costituito** da un insieme di componenti elementari, divisi in tre categorie:

-primari,
-secondari
-e di aiuto.

Di seguito studieremo solo i primari, ovvero:

-elementi semplici
-elementi complessi
-indicatori
-attributi

Uno XML Schema deve avere la seguente **struttura generale**:

```
<?xml version"1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    "definizione della struttura"

</xs:schema>
```

Notare come il tag <xs:schema> specifichi attraverso l'attributo xmlns che in questo documento sarà applicato lo standard del W3C.

**ELEMENTI SEMPLICI**

Gli elementi semplici sono quei componenti che non possono contenere altri elementi o attributi al loro interno, ma solo tipi standard.
Sono elementi semplici

| | |
|---|---|
| -i numeri | (es: int, float, double, byte, ...), |
| -le date | (es: date, duration, ...), |
| -i booleani | (boolean), |
| -gli URI | (uriReference), |
| -le stringhe di testo | (es: string). |

Gli elementi semplici si indicano col tag:

```
<xs:element nomeElemento tipoElemento ...>
```

Si possono inoltre definire valori fissi (fixed) o di default (default);

la differenza è che

-con default il valore viene applicato al contenuto dell'elemento solo se questo è vuoto,

-con fixed l'assegnamento avviene sempre. Alcuni esempi:

```
<xs:element name="Nome1" type="tipo1" />
<xs:element name="Nome2" type="tipo1" default="ciao" />
<xs:element name="Nome3" type="tipo2" fixed  ="bello" />
```

E' infine possibile imporre diversi tipi di restrizioni sui valori, tra cui:

| | |
|---|---|
| -whiteSpace, | tipo di gestione dei whitespace |
| -totalDigit, | numero esatto di caratteri alfanumerici |
| -pattern, | sequenza di caratteri |
| -min/maxLength, | lunghezza minima e massima |
| -min/maxInclusive, | limite numerico inferiore/superiore, esso compreso |
| -min/maxExclusive, | limite numerico inferiore/superiore, esso escluso |
| -length, | numero esatto di caratteri (>=0) |
| -fractionDigit, | numero di decimali dopo la virgola (>=0) |
| -enumeration, | lista di valori accettabili |

**ELEMENTI COMPLESSI**

Gli elementi complessi sono quei componenti dell'XSD che possono contenere attributi o altri elementi al loro interno.

Si indicano col tag:

```
<xs:complexType>
        ...
</xs:complexType>
```

**INDICATORI**

Nel caso in cui l'elemento ne contenga altri, è possibile definire ordinamenti, cardinalità e raggruppamenti attraverso l'uso degli indicatori.

In dettaglio:

indicatori di ordinamento

| | |
|---|---|
| any: | qualunque elemento, in qualunque ordine |
| all: | tutti gli elementi, in qualunque ordine |
| choice: | uno e un solo elemento |
| sequence: | tutti gli elementi, nell'ordine specificato |

indicatori di cardinalità

maxOccurs: max numero di occorrenze (default 1)

minOccurs: min numero di occorrenze (default 1)

raggruppamento

Group name

Group reference

Un esempio di dichiarazione di elemento complesso è il seguente:

```
(1) <xs:element name="persona">
(2)       <xs:complexType>
(3)       <xs:sequence>
(4)             <xs:element name="Cognome"  type="xs:string" />
(5)             <xs:element name="Nome" type="xs:string" maxOccurs="4" minOccurs="1" />
(6)       </xs:sequence>
(7)       </xs:complexType>
(8) </xs:element>
```

Commenti punto per punto:

(1) definisce un nuovo elemento che ha nome "persona"
(2) annuncia che tale elemento è di tipo complesso
(3) dichiara che gli elementi seguenti dovranno essere tutti utilizzati nell'ordine specificato
(4) definisce un nuovo elemento semplice di tipo stringa chiamato "Cognome"
(5) definisce un nuovo elemento semplice di tipo stringa chiamato "Nome", obbligatorio e che può avere massimo 4 occorrenze (es. Andrea Giovanni Battista)
(6), (7) e (8) sono tags di chiusura

Infine, un elemento di tipo complesso può essere a sua volta un elemento complesso. Pensiamo ad esempio a un elemento che contenga i dati di una persona, tra cui anche l'indirizzo, anch'esso composto da più elementi (via, numero, città, ...).
Oltre a permettere una nidificazione che meglio struttura il nostro documento XML, questa proprietà consente di condividere tra più elementi alcuni tipi complessi.

**ATTRIBUTI**

Gli attributi sono quei componenti dell'XSD che forniscono informazioni aggiuntive sugli elementi complessi.
Si indicano col tag:

```
<xs:attribute ... ... />
```

Tra le proprietà che possono essere definite al loro interno ricordiamo senz'altro:

name = "NomeAttributo" per indicare il nome
type = "TipoAttributo" per il tipo, che può essere solo uno di quelli standard, non derivati
use = "TipoDiUso" che specifica se l'attributo è obbligatorio o opzionale
default = "valoreDef", fixed="valoreFisso" e le varie restrizioni, tutte proprietà che abbiamo già visto per gli elementi semplici.

Un esempio di dichiarazione di attributo è il seguente:

```
<xs:attribute name="NomeAttributo1" type="TipoAttributo1" />
<xs:attribute name="NomeAttributo2" type="TipoAttributo1" default|fixed="ciao" use="required|optional" />
```

ESEMPIO XML Schema

```
(1) <?xml version="1.0"?>
(2)<schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
(3)     <xs:element name="mail">
(4)     <xs:complexType>
(5)         <xs:sequence>
(6)             <xs:element name="a"          type="xs:string">
(7)             <xs:element name="da"         type="xs:string">
(8)             <xs:element name="oggetto"    type="xs:string">
(9)             <xs:element name="testo"      type="xs:string">
(10)        </xs:sequence>
(11)    </xs:complexType>
(12)    <xs:attribute name="quando" type="xs:date" use="required" />
(13)    </xs:element>
(14)</schema>
```

Commenti punto per punto:

| | |
|---|---|
| (1) e (2), | indicano la versione e lo standard di riferimento del documento |
| (3) | definisce un nuovo elemento che ha nome "mail" |
| (4) | annuncia che tale elemento è di tipo complesso |
| (5) | dichiara che gli elementi seguenti dovranno essere tutti utilizzati nell'ordine specificato |
| (6) (7), (8) e (9) | definisce quattro nuovi elementi semplici di tipo stringa che compongono l'elemento complesso "mail". Essi si chiamano rispettivamente "a", "da", "oggetto" e "testo" |
| (10) e (11) | sono tags di chiusura |
| (12) | definisce un attributo di nome "quando" per il tipo complesso "nome". E' di tipo "data" ed il suo utilizzo è obbligatorio |
| (13) e (14) | sono tags di chiusura |

Documento XML valido

```
<?xml version="1.0"?>
<mail quando="15/07/2006">
        <a>         Rossi Andrea        </a>
        <da>        Bianchi Riccardo    </da>
        <oggetto>   Promemoria          </oggetto>
        <testo>     Compleanno Michela  </testo>
</mail>
```

Nel documento xml va indicato lo ‘schemaLocation’ del file xsd e l’href del file css. Per esempio:

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/css" href="libro.css" ?>
<mail xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="libro.xsd">

        --- --- ---
</mail>
```

# XSD Elements

| Element | Explanation |
|---|---|
| all | Specifies that the child elements can appear in any order. Each child element can occur 0 or 1 time |
| annotation | Specifies the top-level element for schema comments |
| any | Enables the author to extend the XML document with elements not specified by the schema |
| anyAttribute | Enables the author to extend the XML document with attributes not specified by the schema |
| appinfo | Specifies information to be used by the application (must go inside annotation) |
| attribute | Defines an attribute |
| attributeGroup | Defines an attribute group to be used in complex type definitions |
| choice | Allows only one of the elements contained in the <choice> declaration to be present within the containing element |
| complexContent | Defines extensions or restrictions on a complex type that contains mixed content or elements only |
| complexType | Defines a complex type element |
| documentation | Defines text comments in a schema (must go inside annotation) |
| element | Defines an element |
| extension | Extends an existing simpleType or complexType element |
| field | Specifies an XPath expression that specifies the value used to define an identity constraint |
| group | Defines a group of elements to be used in complex type definitions |
| import | Adds multiple schemas with different target namespace to a document |
| include | Adds multiple schemas with the same target namespace to a document |
| key | Specifies an attribute or element value as a key (unique, non-nullable, and always present) within the containing element in an instance document |
| keyref | Specifies that an attribute or element value correspond to those of the specified key or unique element |
| list | Defines a simple type element as a list of values |
| notation | Describes the format of non-XML data within an XML document |
| redefine | Redefines simple and complex types, groups, and attribute groups from an external schema |
| restriction | Defines restrictions on a simpleType, simpleContent, or a complexContent |
| schema | Defines the root element of a schema |
| selector | Specifies an XPath expression that selects a set of elements for an identity constraint |
| sequence | Specifies that the child elements must appear in a sequence. Each child element can occur from 0 to any number of times |
| simpleContent | Contains extensions or restrictions on a text-only complex type or on a simple type as content and contains no elements |
| simpleType | Defines a simple type and specifies the constraints and information about the values of attributes or text-only elements |

| | |
|---|---|
| union | Defines a simple type as a collection (union) of values from specified simple data types |
| unique | Defines that an element or an attribute value must be unique within the scope |

# XSD Restrictions

| Constraint | Description |
|---|---|
| enumeration | Defines a list of acceptable values |
| fractionDigits | Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero |
| length | Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero |
| maxExclusive | Specifies the upper bounds for numeric values (the value must be less than this value) |
| maxInclusive | Specifies the upper bounds for numeric values (the value must be less than or equal to this value) |
| maxLength | Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero |
| minExclusive | Specifies the lower bounds for numeric values (the value must be greater than this value) |
| minInclusive | Specifies the lower bounds for numeric values (the value must be greater than or equal to this value) |
| minLength | Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero |
| pattern | Defines the exact sequence of characters that are acceptable |
| totalDigits | Specifies the maximum number of digits allowed. Must be greater than zero |
| whiteSpace | Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled |