

# **XML SCHEMA**

---

Engenharia de Dados e Conhecimento

# O que é um XML Schema?

- O objectivo do XML Schema é definir o modelo de dados presente num documento XML.
- A linguagem XML Schema é conhecida como "XML Schema Definition" (XSD).

# A norma XML Schema

- A norma está dividida em 3 partes:
- XML Schema Part 0: **Primer**
  - é um documento não-normativo destinado a fornecer uma descrição de fácil leitura das implicações do XML Schema, e é orientada para a compreensão de como rapidamente para criar esquemas usando a linguagem XML Schema.
- XML Schema Part 1: **Estrutura** e XML Schema Part 2: **Tipos de dados**
  - Documentos que fornecem a descrição normativa completa da linguagem XML Schema.
  - Descrevem as características da linguagem através de numerosos exemplos que são complementados com referências às normas.

# O que é o XML Schema?

- XML Schema é uma norma que define:
  - Os elementos que aparecem num documento
  - Quais são "child elements"
    - A ordem dos diversos "child elements"
    - O número de "child elements"
  - Quando um elemento é "vazio" ou quando pode incluir texto
  - Quais os atributos que podem aparecer num elemento
  - Define ainda os "default" e "fixed values" para elementos e atributos
  - Quais os "data types" para elementos e atributos

# XML Schemas vs DTDs (I)

- Neste momento os XML Schemas já são utilizados na generalidade das aplicações Web em substituição dos DTDs.
- Algumas das razões para isso:
  - são extensíveis e permitem futuras adições
  - são mais ricos e precisos que os DTDs
  - são escritos em XML (um DTD não é!)
  - suportam tipos de dados ("data types") (um DTD só suporta PCDATA e CDATA types)
  - suportam namespaces

# XML Schemas vs DTDs (II)

- Uma das principais vantagens dos XML Schemas é suportarem "data types".
- Com suporte para "data types" é mais fácil:
  - Descrever o que é ou não permitido escrever num documento;
  - Validar a correção dos dados do documento;
  - Manusear dados provenientes de uma base de dados;
  - É fácil definir um conjunto de restrições a aplicar aos dados;
  - É fácil definir padrões de dados;
  - É mais fácil converter dados entre diferentes tipos de dados.

# XML Schemas vs DTDs (III)

Quando são enviados dados entre dois agentes, é essencial que tanto o emissor como o receptor tenham as mesmas "expectativas" relativamente aos conteúdos a transmitir.

- Através dos XML Schemas, o emissor pode descrever os dados de modo que o receptor os perceba.
- Exemplo:
  - Uma data qualquer, como por exemplo: "03-11-2004" pode, em alguns países, ser interpretada como 3 de Novembro e noutros como 11 de Março mas, em XML, um elemento com um tipo de dados deste tipo deverá ser descrito como:

```
<date type="date">2004-03-11</date>
```
  - Isso garante uma mútua percepção do conteúdo porque o XML data type "date" requer o formato YYYY-MM-DD.

# Exemplo (XML → DTD):

- Documento "note.xml"

```
<?xml version="1.0"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

- Documento "note.dtd"

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```



# Exemplo (XSD):

- Documento "note.xsd"

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">
  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element> </xs:schema>
```

- O elemento **note** é do tipo **complex** porque contém outros elementos.
- Os demais elementos (**to**, **from**, **heading**, **body**) são definidos como tipos simples porque não contém outros elementos.

# Validação

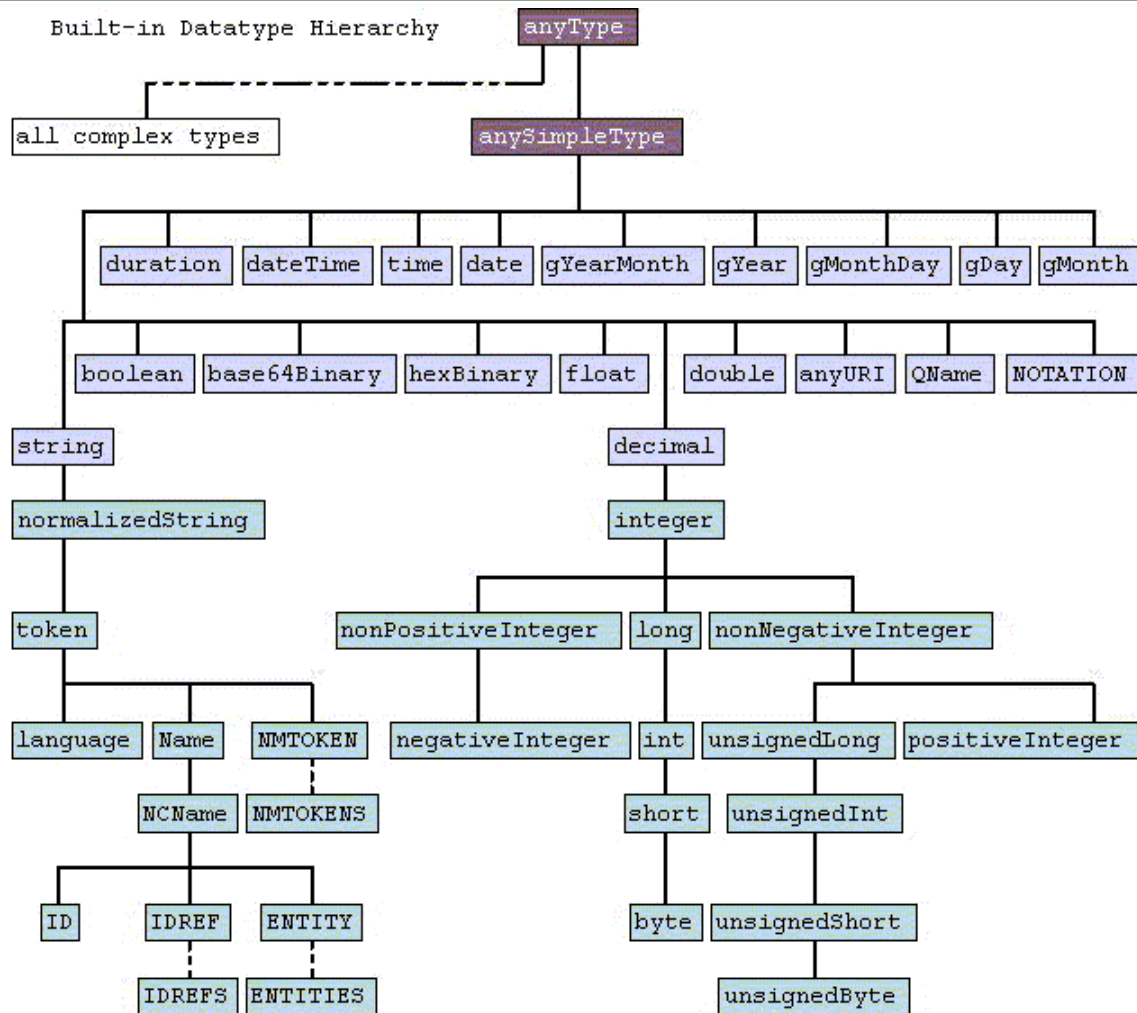
- Os XML Schemas, e os DTDs, podem ser usados para proceder à validação dos dados presentes num documento XML.
- Exemplo de referência de um XML Schema:

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

# **XML SCHEMA PART 2**

---

## **Data Types**



# **XML SCHEMA PART 1**

---

Structures

# XML Schema

- Um XML Schema é composto por **componentes**, tais como definições de tipo e declarações de elementos.
  - Estes são usados para garantir que um documento XML que o incorpora está válido em relação aos seus elementos e atributos
- Um XML Schema especifica ainda acréscimos/aumentos desses itens e dos seus descendentes.
  - Esse aumento de informação torna explícito o que pode ter ficado implícito no documento original, tais como: normalização e/ou atribuição de valores padrão para os (i) elementos, (ii) atributos, (iii) tipos de elementos e (iv) tipos de atributos.

# Componentes de um XML Schema

- Os componentes principais, que podem ou devem existir num documento XML Schema são os seguintes:
  - Tipos simples
  - Tipos complexos
  - Atributos
  - Elementos
- Os componentes secundários são as seguintes:
  - Grupos de atributos
  - Restrições de identidade
  - Grupos de modelos
  - Notações

# XSD – Simple Elements

- Um "simple element" contém só texto.
  - Não pode ter outros elementos ou atributos.
  - O texto pode ser de diferentes tipos:
    - boolean, string, date, etc., ou pode mesmo ser um novo tipo de dados definido pelo utilizador.
    - É também possível adicionar restrições (facets) a um determinado tipo de dados, de modo a limitar o seu conteúdo, e pode ainda ser requerido que os dados estejam de acordo com um determinado padrão pré-definido.
- Sintaxe: 

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

  - Onde xxx representa o nome do elemento e yyy o tipo de dados



# XSD – Simple Elements

- Exemplo:

```
<lastname>Smith</lastname>  
<age>34</age>  
<dateborn>1968-03-27</dateborn>
```

- Definições para o exemplo:

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

# Simple Elements - default & fixed values

- Simple elements podem ter:
  - *default value* - atribuído a um elemento quando nenhum valor é especificado.

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

- *fixed value* - neste caso não é possível atribuir qualquer outro valor

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

# Atributos | *Attributes*

- Se um elemento possuir atributos, é considerado como um elemento complexo.
  - Os *simple elements* não podem possuir atributos.
  - Os *attributes* são definidos como "simple types".
- Exemplo:

```
<lastname lang="EN">Smith</lastname>
```

- E a correspondente definição:

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

# Tipos de atributos

- Todos os atributos são opcionais por defeito.
- Para explicitar se um atributo é obrigatório ou opcional deve utilizar-se o atributo "use":
- Exemplos:

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

- OU

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

# Restrições de valores

- As restrições são para controlar a gama de valores de um determinado elemento ou atributo.
- Às restrições colocadas sobre os elementos XML chamam-se "**facets**".

```
<xs:element name="age">  
  <xs:simpleType>  
    <xs:restriction base="xs:integer">  
      <xs:minInclusive value="0"/>  
      <xs:maxInclusive value="100"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>
```

Define um elemento chamado "age" que pode tomar qualquer valor entre 0 e 100.

# Restrições em grupos enumerados

- No caso do campo não ser do tipo numérico, é também possível enumerar qual a gama de valores que este pode aceitar. Para tal é utilizado o elemento "**enumeration**".
- Exemplo:

```
<xs:element name="car">
  <xs:simpleType>
    < xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="BMW"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- O elemento "car" é um *simple type* com uma restrição
- Os valores permitidos são:
  - Audi, Golf e BMW.

- Este exemplo ...

```
<xs:element name="car">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="BMW"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- ... poderia também ser escrito desta forma:

```
<xs:element name="car" type="carType"/>

<xs:simpleType name="carType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Audi"/>
    <xs:enumeration value="Golf"/>
    <xs:enumeration value="BMW"/>
  </xs:restriction>
</xs:simpleType>
```

Este caso tem a vantagem de o tipo **carType** poder vir a ser reutilizado por outros elementos

# Restrições em séries de valores

- Para limitar o conteúdo de um elemento XML pode ser usada uma expressão regular como valor do elemento `<xs:pattern>`.
- Exemplos:

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

restrição: UMA letra MINÚSCULA no intervalo de a a z.



```
<xs:element name="initials">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="[A-Z][A-Z][A-Z]"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>
```

restrição: TRÊS letras  
MAIÚSCULAS no intervalo  
de A a Z.

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

restrição: TRÊS letras  
MAIÚSCULAS/MINÚSCULAS  
no intervalo de a a z.

```
<xs:element name="choice">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="[xyz]"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>
```

restrição: UMA letra MINÚSCULA x, y  
ou z.

```
<xs:element name="prodid">  
  <xs:simpleType>  
    <xs:restriction base="xs:integer">  
      <xs:pattern value="[0-9][0-9][0-9][0-9][0-9]"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>
```

restrição: uma sequência de  
CINCO dígitos. Todos eles  
deverão estar na gama 0..9

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

Restrição: zero ou mais ocorrências de letras minúsculas do intervalo a a z.

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z][A-Z])+"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrição: UM ou mais pares de letras, cada par com uma letra minúscula e uma maiúscula.

"sToP" será validada por este padrão, mas "Stop", "STOP" ou "stop" não serão.

```
<xs:element name="gender">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="male | female"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrição: valores aceitáveis são male OU female.

# Restrições com espaços em branco

```
<xs:restriction base="xs:string">  
  <xs:whiteSpace value="preserve"/>  
</xs:restriction>
```

whiteSpace = "preserve", diz ao processador para NÃO REMOVER qualquer espaço em branco.

```
<xs:whiteSpace value="replace"/>
```

whiteSpace = "replace", diz ao processador para SUBSTITUIR qualquer espaço em branco (line feeds, tabs, spaces e carriage returns) num espaço.

```
<xs:whiteSpace value="collapse"/>
```

whiteSpace = "collapse", diz ao processador para REMOVER qualquer espaço em branco (line feeds, tabs, spaces e carriage returns) e transformá-los em espaços; os espaços no início e fim dos parágrafos são removidos; múltiplos espaços são convertidos em espaços simples.

# Restrições no tamanho

```
<xs:restriction base="xs:string">  
  <xs:length value="8"/>  
</xs:restriction>
```

Restrição: tipo String, com tamanho exato de 8 carateres

```
<xs:restriction base="xs:string">  
  <xs:minLength value="5"/>  
  <xs:maxLength value="8"/>  
</xs:restriction>
```

Restrição: tipo String, com tamanhos entre 5 e 8 carateres

# Lista de restrições:

enumeration	Define a lista de valores aceitáveis
fractionDigits	especifica o número máximo de casas decimais permitidas. Este valor deve ser maior ou igual a zero
length	Especifica o número exacto de caracteres ou lista de elementos permitido. Este valor deve ser maior ou igual a zero
maxExclusive	Define o limite máximo permitido para um valor (excluindo-o)
maxInclusive	Define o limite máximo para um número
maxLength	Especifica o número máximo de caracteres ou lista de elementos permitido. Este valor deve ser maior ou igual a zero
minExclusive	Define o limite mínimo permitido para um valor (excluindo-o)
minInclusive	Define o limite mínimo para um número
minLength	Especifica o número mínimo de caracteres ou lista de elementos permitido. Este valor deve ser maior ou igual a zero
pattern	Define exactamente a sequência de caracteres permitidos
totalDigits	Especifica o número exacto de dígitos permitidos. Tem de ser maior que zero.
whiteSpace	Especifica o modo como os caracteres (line feeds, tabs, espaços e carriage returns) são tratados

# XSD Complex Elements

- Um elemento XML complexo contém outros elementos e/ou atributos.
- Há quatro tipos de elementos complexos:
  - vazios
  - que contêm somente outros elementos
  - que contêm somente texto
  - que contêm outros elementos e texto
- Qualquer destes elementos pode possuir atributos

# XSD Complex Elements

- Elemento vazio:

```
<product pid="1345"/>
```

- Elemento com outros elementos:

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



# XSD Complex Elements

- Contendo somente texto:

```
<food type="dessert">Ice cream</food>
```

- Contendo texto e outros elementos:

```
<description>  
It happened on <date lang="norwegian">03.03.99</date> ....  
</description>
```

# Definição de elementos complexos

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



```
<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>
```

os *child elements* estão numa  
**<sequence>** → devem aparecer  
na ordem em que foram  
declarados.

# Definição de tipos complexos

- O elemento "employee" pode ser definido à custa da definição prévia de um tipo complexo:

```
<xs:element name="employee" type="personinfo"/>

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

# Definição de elementos complexos

- Vários elementos podem ser definidos à custa do mesmo tipo complexo:

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>

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

# Definição de elementos complexos

- É possível basear um tipo complexo noutro já existente e adicionar-lhe mais elementos

```
<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

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

# XSD Complex Types Indicators

- É possível controlar como os elementos podem ser utilizados nos documentos através dos *indicators*
- A indicação dos tipos complexos pode tomar os seguintes valores:
  - ALL
  - CHOICE
  - SEQUENCE

# XSD Complex Types Indicators

- All Indicator
  - O indicador `<all>` especifica que os child elements **podem aparecer em qualquer ordem** e que cada *child element* deverá aparecer unicamente uma vez.

```
<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>
```

# XSD Complex Types Indicators

- Choice Indicator

- O indicador <choice> especifica que **qualquer um** dos elementos pode ocorrer.

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice>
      <xs:element name="employee" type="employee"/>
      <xs:element name="member" type="member"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```



# XSD Complex Types Indicators

- Sequence Indicator
  - O indicador <sequence> especifica que os child elements **devem aparecer na ordem/sequência indicada**

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

# XSD – Occurrence Indicators

- Indicadores de ocorrência
  - são utilizados para definir quantas vezes um elemento pode ocorrer.
    - maxOccurs Indicator
    - minOccurs Indicator
  - O default value para estes elementos é 1, ou seja, se nada for dito, ocorre sempre uma vez.

# XSD – Group Indicators

- Indicadores de grupo
  - Os indicadores de grupo (<group>) são utilizados para relacionar grupos de elementos

```
<xs:group name="personGroup">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>

<xs:element name="person" type="personByCountry"/>

<xs:complexType name=" personByCountry ">
  <xs:sequence>
    <xs:group ref="personGroup"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

# XSD - `<any>` Element

- O elemento `any`
  - O elemento `<any>` permite acrescentar ao documento XML elementos não especificados no schema.

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
      <xs:any minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

O exemplo mostra a declaração do elemento **"person"**.

Através da utilização do elemento `<any>` podemos acrescentar qualquer informação depois do elemento `<lastname>`.

- Exemplo

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<persons xmlns="http://www.microsoft.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:SchemaLocation="http://www.microsoft.com family.xsd
  http://www.w3schools.com children.xsd">
  <person>
    <firstname>Hege</firstname>
    <lastname>Refsnes</lastname>
    <children>
      <childname>Cecilie</childname>
    </children>
  </person>
  <person>
    <firstname>Stale</firstname>
    <lastname>Refsnes</lastname>
  </person>
</persons>
```

# XSD - `<anyAttribute>` Element

- O elemento `anyAttribute`
  - O `<anyAttribute>` permite acrescentar atributos não especificados no schema.

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

O exemplo mostra uma declaração do elemento **"person"**.

Através da utilização do elemento `<anyAttribute>` pode-se adicionar qualquer número de atributos aos elementos de **"person"**.

- Exemplo

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<persons xmlns="http://www.microsoft.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:SchemaLocation="http://www.microsoft.com family.xsd
  http://www.w3schools.com children.xsd">
  <person gender="female">
    <firstname>Hege</firstname>
    <lastname>Refsnes</lastname>
  </person>
  <person gender="male">
    <firstname>Stale</firstname>
    <lastname>Refsnes</lastname>
  </person>
</persons>
```