

XML Schema

Agenda

- XSD – XML Schema
 - Elementos
 - Atributos
 - Restrições
 - Tipos de dados

- Mesmo estando bem formados, os documentos podem conter erros de **validação** considerando a âmbito em que estão inseridos.
- Um Schema XML descreve a **estrutura** e o **vocabulário** de um documento XML;
- O principal propósito passa por definir as **regras** de construção de um documento XML:
 - Os **elementos** e **atributos** que podem surgir no documento;
 - O **número** e a **ordem** dos elementos filho;
 - **Tipos** de dados para elementos e atributos;
 - (...)

- Através da utilização de um **parser**, é possível **validar** documentos XML;
- Além disso, o XSD é escrito utilizando **XML**;
- Com XSD, Podemos:
 - **Reutilizar** os schemas;
 - **Criar** os nossos próprios tipos de dados;
 - **Referenciar** múltiplos schemas num documento

Introdução

- O elemento `schema` é o `root` de qualquer documento XSD;
- O elemento `schema` contém a declaração do `namespace` XSD:

```
<?xml version="1.0" encoding="UTF-8"?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

...

```
</xs:schema>
```

- Documentos XSD têm a extensão `.XSD`.

Namespaces

- **Namespaces** representam uma forma de **evitar** conflitos de nomes;
- Como os elementos XML são definidos por quem os cria, diversos **conflitos** podem surgir quando se misturam documentos de diferentes áreas de aplicação;

Namespaces

- O fragmento: `xmlns:xs="http://www.w3.org/2001/XMLSchema"`
- Indica que os elementos e tipos de dados do schema são definidos no namespace:
`http://www.w3.org/2001/XMLSchema`
- Também indica que os **elementos** e **tipos** de dados desse namespace devem ser precedidos do **prefixo**: `xs`;

Simple elements

- Um **elemento simples** é um elemento XML que contém apenas texto;
- **Não** pode conter outros elementos ou atributos.
- No entanto, o “texto” pode ser de muitos **tipos diferentes** incluídos na definição do XML Schema (booleano, string, data, etc.), ou pode ser um tipo personalizado.
- Podem ser adicionadas **restrições** (facetas) para limitar seu conteúdo, o que faz com que os dados correspondam a um padrão específico.

Simple elements

- Exemplo:

```
(...)  
<xs:element name= "nome" type= "xs:string"/>  
(...)
```

Simple elements

- O XSD possui vários **tipos** de dados. Os tipos mais comuns são:

Tipos de dados	Exemplo (XSD)	Exemplo (XML)
xs:string	<code><xs:element name="cliente" type="xs:string"/></code>	<code><cliente>Augusto Pinto</cliente></code>
xs:decimal	<code><xs:element name="preco" type="xs:decimal"/></code>	<code><preco>25.38</preco></code>
xs:integer	<code><xs:element name="idade" type="xs:integer"/></code>	<code><idade>25</idade></code>
xs:boolean	<code><xs:attribute name="stock" type="xs:boolean"/></code>	<code><livro stock="true">Os Maias</livro></code>
xs:date	<code><xs:element name="data" type="xs:date"/></code>	<code><data>2021-10-13</data></code>
xs:time	<code><xs:element name="inicio" type="xs:time"/></code>	<code><inicio>09:23:40</inicio></code>
xs:dateTime	<code><xs:element name="data_inicio" type="xs:dateTime"/></code>	<code><data_inicio>2021-10-13T09:23:40</data_inicio></code>

- Outras variações podem ser utilizadas:

int

Números inteiros positivos/negativos de 32-bit

long

Números inteiros positivos/negativos de 64-bit

short

Números inteiros positivos/negativos de 16-bit

negativeInteger

Inteiros contendo números negativos

nonNegativeInteger

Inteiros apenas com números positivos

unsignedLong

Números inteiros positivos de 64-bit

unsignedInt

Números inteiros positivos de 32-bit

unsignedShort

Números inteiros positivos de 16-bit

Simple elements – Facets

- Um valor **fixo** é atribuído ao elemento e mais nenhum valor pode ser especificado:

```
(...)  
<xs:element name="cor" type="xs:string" fixed="vermelho"/>  
(...)
```

Simple elements – Facets

- **Restringir** o valor da “idade” para que esteja contido no intervalo de 0 e 120:

(...)

```
<xs:element name="idade">
```

```
  <xs:simpleType>
```

```
    <xs:restriction base="xs:integer">
```

```
      <xs:minInclusive value="0"/>
```

```
      <xs:maxInclusive value="120"/>
```

```
    </xs:restriction>
```

```
  </xs:simpleType>
```

```
</xs:element>
```

(...)

Simple elements – Facets

- Para **limitar** o conteúdo de um elemento XML a uma série de números ou letras, podemos utilizar expressões regulares;
- Exemplo: Aceitar apenas **letras minúsculas** de a a z:

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

Simple elements – Facets

- Exemplo: 3 letras maiúsculas de a a z:

(...)

```
<xs:restriction base="xs:string">
```

```
<xs:pattern value="[A-Z][A-Z][A-Z]"/>
```

```
</xs:restriction>
```

(...)

- Exemplo: 3 letras maiúsculas ou minúsculas de a a z: (...)

```
<xs:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>
```

(...)

- Exemplo: As letras x, y e z são as únicas permitidas: (...)

```
<xs:pattern value="[xyz]"/>
```

(...)

Simple elements – Facets

- Exemplo: 5 dígitos de 0 a 9:

```
(...)  
<xs:pattern value="[0-9][0-9][0-9][0-9][0-9]"/>  
(...)
```

- Exemplo: 0 ou mais ocorrências de letras de a a z:

```
(...)  
<xs:pattern value="([a-z])*"/>  
(...)
```

- Exemplo: Pares de letras com minúscula e maiúscula:

```
(...)  
<xs:pattern value="([a-z][A-Z])+"/>  
(...)
```


Simple elements – Facets

- Exemplo: Conjunto de valores:

(...)

```
<xs:pattern value="male|female"/>
```

(...)

- Exemplo: 8 caracteres que podem ser minúsculas ou maiúsculas de a a z ou número de 0 a 9:

(...)

```
<xs:pattern value="[a-zA-Z0-9]{8}"/>
```

(...)

Simple elements – Facets

- Exemplo: Validação de um email:

```
<xs:pattern value="([A-Za-z0-9._%+-]+)@([A-Za-z0-9-]+\.[A-Za-z]{2,})"></xs:pattern>
```

The diagram illustrates the components of the email validation regex. The regex is: `<xs:pattern value="([A-Za-z0-9._%+-]+)@([A-Za-z0-9-]+\.[A-Za-z]{2,})"></xs:pattern>`. It is divided into three main sections by green boxes: 1. The local part: `([A-Za-z0-9._%+-]+)`. 2. The @ symbol: `@`. 3. The domain part: `([A-Za-z0-9-]+\.[A-Za-z]{2,})`. Blue arrows point from each section to its corresponding explanation below.

Permite um ou mais dos seguintes caracteres na parte local (antes do @):

- Letras maiúsculas e minúsculas (A-Za-z)
- Dígitos (0-9)
- Pontos (.), sublinhados (_), sinais de percentagem (%), sinais de mais (+) e hífen (-), todos eles válidos em endereços de e-mail.

O símbolo literal @

Permite um ou mais caracteres na parte do domínio (após o @):

- Letras maiúsculas e minúsculas (A-Za-z)
- Dígitos (0-9)
- Pontos (.) e hífen (-) são permitidos nos nomes de domínio.

impõe que o domínio deve ter um domínio que consiste em pelo menos duas letras (por exemplo, .com, .org, etc.). O {2,} significa que deve ter pelo menos dois caracteres, mas pode ser mais longo (por exemplo, .info, .museum).

Simple elements – Facets

- Para limitar o tamanho de um valor, podemos utilizar as restrições: maxLength e minLength;

```
(...)  
<xs:length value="8"/>  
(...)
```

- Exemplo: mínimo de 5 e máximo de 8 caracteres

```
(...)  
<xs:minLength value="5"/>  
<xs:maxLength value="8"/>  
(...)
```

Complex Elements - Atributos

- Se um elemento contém **atributos**, é considerado um elemento **complexo**. No entanto, o atributo é declarado como um tipo **simples**;

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

- Valores **fixos** ou por **defeito** podem também ser configurados:

```
(...)
```

```
<xs:attribute name="title" type="xs:string" default="EN"/>
```

```
(...)
```

```
(...)
```

```
<xs:attribute name="title" type="xs:string" fixed="EN"/>
```

```
(...)
```

Complex Elements - Atributos

Os atributos são **opcionais** por defeito. Para que sejam obrigatórios, é necessário utilizar o atributo “**use**”:

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

Complex Elements

Exemplo completo com atributos:

```
<xs:complexType name="emailType">
  <xs:sequence>
    <xs:element name="to" type="xs:string" minOccurs="1" maxOccurs="1"/>
    (...)
    <xs:element name="body" type="xs:string" minOccurs="1" maxOccurs="1"/>
  </xs:sequence>

  <xs:attribute ref="type" use="required"/>

</xs:complexType>
```

Complex Elements

Um elemento complexo é um elemento XML que contém outros elementos e/ou atributos;

Existem 4 tipos:

- Elementos vazios: `<elementoVazio/>`
- Elementos que contêm outros elementos:

```
<pai>  
  <filho1></filho1>  
  <filho2></filho2>  
</pai>
```

- Elementos que apenas contêm texto:

```
<elementoTexto atributo="valor">Algum texto aqui</elementoTexto>
```

- Elementos com elementos e texto.

```
<elementoComplexo>Texto aqui.<filho></filho></elementoComplexo>
```

Complex Elements

Duas formas de construir elementos complexos:

1. Na especificação do elemento:

(...)

```
<xs:element name="pessoa">
```

```
  <xs:complexType>
```

```
    <xs:sequence>
```

```
      <xs:element name="primeiroNome" type="xs:string"/>
```

```
      <xs:element name="ultimoNome" type="xs:string"/>
```

```
    </xs:sequence>
```

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

```
</xs:element>
```

(...)

Indicador de ordem (preserva a ordem dos elementos)

Apenas o elemento pessoa pode utilizar este elemento complexo

Duas formas de construir elementos complexos:

2. Utilizar o atributo type que refere o elemento que utiliza o elemento complexo:

(...)

```
<xs:element name="pessoa" type="infoPessoa"/>
```

```
<xs:complexType name="infoPessoa">
```

```
  <xs:sequence>
```

```
    <xs:element name="primeiroNome" type="xs:string"/>
```

```
    <xs:element name="ultimoNome" type="xs:string"/>
```

```
  </xs:sequence>
```

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

(...)

Vários elementos podem utilizar este elemento complexo

Complex Elements

- Um tipo complexo misto pode conter atributos, elementos e texto:

(...)

```
<xs:element name="letter" type="lettertype"/>
```

```
<xs:complexType name="lettertype" mixed="true">
```

```
  <xs:sequence>
```

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

```
    <xs:element name="orderid" type="xs:positiveInteger"/>
```

```
    <xs:element name="shipdate" type="xs:date"/>
```

```
  </xs:sequence>
```

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

(...)

<letter>

Dear Mr.<name>John Smith</name>.

Your order <orderid>1032</orderid>

will be shipped on <shipdate>2001-07-13</shipdate>

</letter>

Complex Elements: `xs:simpleContent` e `xs:extension`

- O elemento `simpleContent` contém extensões num tipo complexo de texto ou num tipo simples, não contendo elementos;
- Tem como elemento pai: `xs:complexType`;
- O elemento `extension` estende um tipo existente (`simpleType` ou `complexType`);
- Tem como elementos pai: `xs:simpleContent` ou `xs:complexContent`;

Complex Elements: xs:simpleContent e xs:extension

- Extensão de um elemento:

XML

(...)

```
<email type = "Profissional">(...)</email>
```

(...)

XSD

(...)

```
<xs:element name="email">
```

```
<xs:complexType>
```

```
<xs:simpleContent>
```

```
<xs:extension base="emailType">
```

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

```
</xs:extension>
```


```
</xs:simpleContent>
```

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

```
</xs:element>
```

(...)

Extension do elemento
emailType



Complex Elements: xs:simpleContent e xs:extension

- Exemplo de um elemento que apenas contém texto e atributos:

```
(...)  
<xs:element name="shoesize" type="shoetype"/>  
  
<xs:complexType name="shoetype">  
  <xs:simpleContent>  
    <xs:extension base="xs:string">  
      <xs:attribute name="country" type="xs:string" />  
    </xs:extension>  
  </xs:simpleContent>  
</xs:complexType>  
(...)
```

Complex Elements: Indicadores

- Os **indicadores de ordem** permitem restringir a ordem com que os elementos surgem no documento;
- Indicadores de ordem:
 - All
 - Choice
 - Sequence
 - Indicadores de ocorrência:
 - maxOccurs
 - minOccurs

Complex Elements: Indicadores

- O indicador: `<all>` especifica que os elementos filho podem surgir em qualquer ordem mas cada um deve surgir apenas uma vez:

```
(...)  
<xs:element name="pessoa">  
  <xs:complexType>  
    <xs:all>  
      <xs:element name="primeiroNome" type="xs:string"/>  
      <xs:element name="ultimoNome" type="xs:string"/>  
    </xs:all>  
  </xs:complexType>  
</xs:element>  
(...)
```

Complex Elements: Indicadores

- O indicador: `<choice>` especifica que qualquer um dos elementos pode ocorrer:

```
(...)  
<xs:choice>  
  <xs:element name="cliente" type="cliente"/>  
  <xs:element name="socio" type="socio"/>  
</xs:choice>  
(...)
```


Complex Elements: Indicadores

- O indicador: `<sequence>` especifica que os elementos filho devem surgir numa ordem específica:

```
(...)  
<xs:element name="pessoa">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="primeiroNome" type="xs:string"/>  
      <xs:element name="ultimoNome" type="xs:string"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>  
(...)
```

Complex Elements: Indicadores

- Os indicadores de ocorrência determinam o número de ocasiões que um elemento pode ocorrer;
- O indicador: `<maxOccurs>` determina o número máximo de ocorrências:

```
(...)  
<xs:sequence>  
  <xs:element name="nomeCompleto" type="xs:string"/>  
  <xs:element name="nomeFilho" type="xs:string" maxOccurs="10"/>  
</xs:sequence>  
(...)
```

Complex Elements: Indicadores

- O indicador: `<minOccurs>` determina o número mínimo de ocorrências:

```
(...)  
<xs:sequence>  
  <xs:element name="nomeCompleto" type="xs:string"/>  
  <xs:element name="nomeFilho" type="xs:string" minOccurs="0"/>  
</xs:sequence>  
(...)
```

Complex Elements: Indicadores

- A configuração: `maxOccurs="unbounded"` permite que o elemento ocorra infinitamente.

```
(...)  
<xs:sequence>  
  <xs:element name="nomeCompleto" type="xs:string"/>  
  <xs:element name="nomeFilho" type="xs:string" maxOccurs="unbounded"/>  
</xs:sequence>  
(...)
```

Associação de um documento XSD a um documento XML

```
<?xml version="1.0" encoding="UTF-8"?>
<peessoas xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xsi:noNamespaceSchemaLocation="pessoaTypes.xsd">
  <pessoa>
    <primeiroNome>Pedro</primeiroNome>
    <ultimoNome>Pinto</ultimoNome>
  </pessoa>
  <pessoa>
    <primeiroNome>Fernando</primeiroNome>
    <ultimoNome>Santos</ultimoNome>
  </pessoa>
</peessoas>
```

XML Schema