XML Schema

- Um XML schema descreve a estrutura de um documento XML.

- XML Schema é uma linguagem que também costuma ser designada por XML Schema Definition (XSD).

- XML Schema é uma alternativa aos DTDs.

XML Schema é um sucessor do DTD

- XML Schemas são mais poderosos que os DTDs.

- Schemas são escritos em XML.

- Schemas permitem ter tipos de dados.

Documento XML

```
<?xml version="1.0" encoding="ISO-88591"?>
<recado>
  <de>Maria</de>
  <para>Zé</para>
  <mensagem>Cinema às 9:00</mensagem>
</recado>
```

DTD

```
<!ELEMENT recado (de, para, mensagem)>
<!ELEMENT de (#PCDATA)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT mensagem (#PCDATA)>
```

Schema

```
<?xml version="1.0" encoding="iso-88591"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="recado">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="de" type="xs:string"/>
        <xs:element name="para" type="xs:string"/>
        <xs:element name="mensagem" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>

</xs:schema>
```

Elementos Simples e Compostos

- Em XML Schema existem elementos simples e elementos compostos.

- Um elemento simples não pode conter outros elementos ou atributos.

- Um elemento composto pode.

Elementos simples

- Exemplos:

```
<apelido>Silva</apelido>
<idade>34</idade>
<datanasc>1968-03-27</datanasc>
```

- Definições dos elementos:

```
<xs:element name="apelido" type="xs:string"/>
<xs:element name="idade" type="xs:integer"/>
<xs:element name="datanasc" type="xs:date"/>
```

Tipos de dados mais comuns em XML Schema

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Elementos podem ter valores por defeito e valores fixos

- Se nada for especificado, o atributo "cor" fica com o valor "vermelho".

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

- Não se pode mudar um valor que seja definido como "fixed".

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

Atributos

- Exemplo de elemento XML com um atributo:

```
<apelido idioma="PT">Silva</apelido>
```

- Definição:

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

Atributos também podem ter valores por defeito e valores fixos

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

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

Atributos podem ser opcionais ou mandatórios

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

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

Restrições

- Pode-se especificar restrições para os valores dos elementos ou atributos.

- Exemplo: O valor do elemento "idade" tem de estar compreendido entre 0 e 100 inclusive.

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

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

</xs:element>
```

Restrições (cont.)

- Pode-se restringir os valores a um conjunto enumerado.

- Ex: Só aceita os valores Audi, Golf, BMW.

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

<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>
```

Também podíamos ter escrito assim ...

```
<xs:element name="carro" type="Tcarro"/>

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

Restrições através de um padrão

- Exemplo: definir o elemento "letra" que apenas deve permitir uma e uma só letra minúscula.

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

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

</xs:element>
```

Restrições através de um padrão (cont.)

- Só aceita telefones começados por 289, seguidos de 1 traço, seguidos de 6 dígitos.

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

<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:pattern value="289-[0-9]{6}"/>
  </xs:restriction>
</xs:simpleType>

</xs:element>
```

Restrições através de um padrão (cont.)

- Só aceita zero ou mais ocorrências de letras minúsculas.

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

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

</xs:element>
```

Restrições através de um padrão (cont.)

- Só aceita “m” ou “f”.

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

<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:pattern value="m|f"/>
  </xs:restriction>
</xs:simpleType>

</xs:element>
```

Restrições de comprimento

- O valor tem de ter no mínimo 5 e no máximo 8 caracteres.

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

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

</xs:element>
```

Elementos compostos

- Um elemento composto pode conter outros elementos e/ou atributos.

- Existe 4 tipos de elementos compostos:
  - elementos vazios
  - elementos que só contêm outros elementos
  - elementos que só contêm texto
  - elementos que contêm outros elementos e também texto

Exemplos de elementos compostos

Ex 1:

```
<produto id="1345"/>
```

Ex 2:

```
<empregado>
   <nomeproprio>Pedro</nomeproprio>
   <apelido>Lopes</apelido>
</empregado>
```

Definição de um elemento composto (1)

- Declaração directa:

```
<xs:element name="empregado">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nomeproprio"
                  type="xs:string"/>
      <xs:element name="apelido"
                  type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Definição de um elemento composto (2)

- Através de um atributo "type":

```
<xs:element name="empregado"
            type="Tpessoa"/>

<xs:complexType name="Tpessoa">
  <xs:sequence>
    <xs:element name="nomeproprio"
                type="xs:string"/>
    <xs:element name="apelido"
                type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Definição de um elemento composto (3)

- Vários elementos podem ter o mesmo tipo (como nas linguagens de programação).

```
<xs:element name="empregado" type="Tpessoa"/>
<xs:element name="estudante" type="Tpessoa"/>
<xs:element name="membro" type="Tpessoa"/>

<xs:complexType name="Tpessoa">
  <xs:sequence>
    <xs:element name="nomeproprio"
                type="xs:string"/>
    <xs:element name="apelido"
                type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Definição de um elemento composto (4)

- Pode-se utilizar um mecanismo de herança (como em POO).

```
<xs:element name="empregado"
            type="Tpessoa2"/>

<xs:complexType name="Tpessoa">
  <xs:sequence>
    <xs:element name="nomeproprio"
                type="xs:string"/>
    <xs:element name="apelido"
                type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

(cont.)

```
<xs:complexType name="Tpessoa2">
  <xs:complexContent>
    <xs:extension base="Tpessoa">
      <xs:sequence>
        <xs:element name="morada"
                    type="xs:string"/>
        <xs:element name="cidade"
                    type="xs:string"/>
        <xs:element name="país"
                    type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Elementos vazios

Ex:

```
<produto id="1345" />
```

Def:

```
<xs:element name="produto">
  <xs:complexType>
    <xs:attribute name="id"
                  type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```

Elementos que só contêm elementos

```
<pessoa>
<nomeproprio>Luís</nomeproprio>
<apelido>Figo</apelido>
</person>
```

- Definição do elemento "pessoa"

```
<xs:element name="pessoa">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nomeproprio"
                  type="xs:string"/>
      <xs:element name="apelido"
                  type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Elementos só com texto

- Pode conter atributos e texto.

```
<shoesize country="france">35</shoesize>
```

- Definição:

```
<xs:element name="shoesize" type="shoetype"/>

<xs:complexType name="shoetype">
  <xs:simpleContent>
    <xs:extension base="xs:integer">
      <xs:attribute name="country"
                    type="xs:string" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Elementos de conteúdo misto

- Pode conter atributos, elementos, e texto.

```
<letter>
Dear Mr.<name>John Smith</name>.
Your order <orderid>1032</orderid> will be
shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

Elementos de conteúdo misto (cont.)

- Definição:

```
<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>
```

Indicadores

- Indicadores de ordem:
  - All
  - Choice
  - Sequence
- Indicadores de ocorrência:
  - maxOccurs
  - minOccurs

Indicador “all”

- <all> especifica que os elementos filho podem aparecer por qualquer ordem e que cada elemento filho pode ocorrer no máximo uma vez.

```
<xs:element name="pessoa">
  <xs:complexType>
    <xs:all>
      <xs:element name="nomeproprio"
                  type="xs:string"/>
      <xs:element name="apelido"
                  type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

Indicador "choice"

- <choice> especifica que apenas um dos elementos filho pode ocorrer.

```
<xs:element name="pessoa">
  <xs:complexType>
    <xs:choice>
      <xs:element name="empregado"
                  type="Tempregado"/>
      <xs:element name="membro"
                  type="Tmembro"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Indicador “sequence”

- <sequence> especifica que cada elemento filho deve ocorrer pela ordem indicada.

```
<xs:element name="pessoa">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nomeproprio"
                  type="xs:string"/>
      <xs:element name="apelido"
                  type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Indicadores de ocorrência

- maxOccurs: especifica o número máximo de vezes que o elemento ocorre.

- minOccurs: especifica o número mínimo de vezes que o elemento ocorre.

Exemplo: “myfamily.xml”

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<pessoas
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="family.xsd">

  <pessoa>
    <nome>José Silva</nome>
    <nome_filho>Teresa</nome_filho>
  </pessoa>

  <pessoa>
    <nome>Teresa Silva</nome>
    <nome_filho>Maria</nome_filho>
    <nome_filho>Rui</nome_filho>
    <nome_filho>Joana</nome_filho>
    <nome_filho>Simão</nome_filho>
  </pessoa>

  <pessoa>
    <nome>Joana Silva</nome>
  </pessoa>

</pessoas>
```

Schema file: “family.xsd”

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">

<xs:element name="pessoas">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="pessoa" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="nome" type="xs:string"/>
            <xs:element name="nome_filho" type="xs:string"
                        minOccurs="0" maxOccurs="5"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```