Bruno Oliveira
2018/2019

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

P.PORTO

DTD – Document type definition
Introdução a XSD – XML Schema

Processamento Estruturado de Informação

Tópicos a abordar

- DTD – Document Type Definition
  - Elementos e atributos
- XSD – XML Schema
  - Elementos
  - Atributos
  - Restrições
  - Tipos de dados

Quando bem formado não é suficiente

- Um documento XML bem formado é um documento que cumpre as regras de sintaxe do XML (deve iniciar com a declaração XML, todas as *tags* devem ser corretamente encerradas, etc).
- Mesmo estando bem formados, os documentos podem conter erros considerando a âmbito em que estão inseridos;
- O DTD define a estrutura dos elementos e atributos permitidos num documento XML;

Introdução DTD

- Com o DTD, diferentes entidades podem comunicar tendo por base um conjunto de regras comuns;
- As aplicações utilizam DTD para verificar se o documento é válido;
- Exemplo:

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

valida

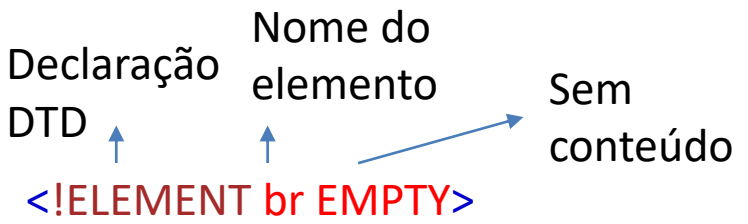
```
<?xml version="1.0"?>
<!DOCTYPE email SYSTEM “mail.dtd">
<email>
  <to>Tove@gmail.com</to>
  <from>Jani@gmail.com</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</email>
```

Introdução DTD

- Do ponto de vista do DTD todos os documentos XML são constituídos por blocos:
  - Elementos: `<body>some text</body>`
  - Attributos: `<img src="computer.gif" />`
  - Entities: Caracteres com significado especial (<,>, etc)
  - PCDATA: Dados/caracteres que podem ser processados por um *parser* e que são colocados entre a tag de inicio e fim do XML.
  - CDATA: Representa os dados/caracteres que não serão processados pelo *parser* e como tal o texto existente não será interpretado como marcação ou caracteres especiais;

Introdução DTD

- Exemplos:
  - Elementos sem conteúdo:



  - Elementos apenas com caracteres interpretados pelo *parser* são declarados como #PCDATA:

```
<!ELEMENT from (#PCDATA)>
```

  - Elementos com um ou mais filhos são declarados com o nome dos filhos entre parêntesis:

```
<!ELEMENT email (to,from,heading,body)>
```

Introdução DTD

- A declaração de atributos tem a seguinte sintaxe:

```
<!ATTLIST element-name attribute-name attribute-type attribute-value>
```

- Por exemplo:

```
<!ATTLIST payment type CDATA "check">
```

Elemento — Atributo — Tipo de bloco — Valor do atributo

Exercício

- Considere o seguinte documento que descreve um conjunto de emails:

```
<?xml version="1.0" encoding="UTF-8"?>
<emails>
    <email type="Pessoal">
        <to>Tove</to>
        <from>Jani</from>
        <heading>Reunião</heading>
        <date>2017/10/02</date>
        <body>Olá, tudo bem?</body>
    </email>
    <email type="Profissional">
        <to>Tove</to>
        <from>Jani</from>
        <heading>Reunião</heading>
        <date>2017/11/02</date>
        <body>Reunião amanhãbody>
    </email>
</emails>
```

Exercício

- Crie um DTD que garanta as seguintes restrições sobre o documento:
  - Cada email contém como elementos filho: to, from, heading, date (opcional) e body;
  - A ordem dos elementos é aquela que se encontra definida no ponto anterior;
  - O atributo type é obrigatório.
  - O valor do atributo *type* deverá ser “Profissional” ou “Pessoal”;

Exemplo

```
<!ELEMENT emails (email*)>
<!ELEMENT email (to,from,heading,date?,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT body (#PCDATA)>
 <!ATTLIST email type (Pessoal|Profissional) #REQUIRED >
```

O elemento emails pode conter 0 ou mais vezes o elemento email

Elementos que são filhos do elemento email, sendo date opcional

Atributo type obrigatório e incluído na tag email

Nota: As ferramentas: Oxygen e Liquid Studio têm suporte para validação DTD

Introdução – XML Schema

- Um Schema XML descreve a estrutura 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;
  - Valores fixos ou por defeito para elementos e atributos.

Introdução – XML Schema

- O *XML Schema (XSD)* é baseado em XML e uma alternativa mais poderosa ao DTD;
- Uma das maiores vantagens do XSD passa pelo suporte para tipos de dados;
- É mais simples descrever o conteúdo do documento;
- É mais simples validar a consistência dos dados, definir regras, formatos de dados e converter entre diferentes tipos;

Introdução – XML Schema

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

Os XML *Schemas* garantem consistência

- Quando dados são trocados entre entidades, é essencial que ambas as partes tenham a mesma "perspetiva" em relação ao conteúdo;
- Com *Schemas* XML, as mensagens podem ser trocadas de forma a que ambas as partes a entendam da mesma forma;
- Por exemplo, a data: "03-11-2017" pode ser interpretada de diferentes formas;
- No entanto, um elemento XML com um tipo de dados, assegura e uniformiza a comunicação;

Documentos XSD

- O elemento schema é o root de qualquer documento XSD:

```
<?xml version="1.0"?>
<xs:schema>
...
</xs:schema>
```

Podemos também acrescentar atributos como a declaração do namespace XSD:

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

- Documentos XSD têm a extensão .XSD.

O que é um namespace?

- 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;
- 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;

Elementos simples XSD

- Um elemento simples contem apenas texto no seu conteúdo, não podendo conter elementos e atributos;
- O "texto" pode ser de diferentes tipos incluídos no XML schema (boolean, string, etc) ou pode ser de um tipo personalizado;
- Também podemos adicionar restrições (facets) para limitar o seu conteúdo.

Elementos simples XSD

- Exemplo: `<xs:element name="xxx" type="yyy"/>`
  - onde **xxx** é o nome do elemento e **yyy** é o tipo de dados do elemento;
- O XSD possui vários tipos de dados. Os tipos mais comuns são:
  - xs:string – mais informação: https://www.w3schools.com/xml/schema_dtypes_string.asp
  - xs:decimal e xs:integer - mais informação: https://www.w3schools.com/xml/schema_dtypes_numeric.asp
  - xs:boolean – mais informação: https://www.w3schools.com/xml/schema_dtypes_misc.asp
  - xs:date e xs:time – mais informação: https://www.w3schools.com/xml/schema_dtypes_date.asp

Tipos de dados: datas e horas

- Uma data é definida como: "YYYY-MM-DD“, onde:
  - YYYY indica o ano;
  - MM indica o mês;
  - DD indica o dia.
- Também é possível especificar time zones: `<start>2002-09-24Z</start>`
- Também é possível especificar a hora no formato: "hh:mm:ss“;

Data e hora – tipo datetime

- O tipo dateTime é utilizado para especificar uma data e uma hora, possuindo o seguinte formato:"YYYY-MM-DDThh:mm:ss“ (nem todos os componentes são obrigatórios) onde:
  - YYYY indica o ano;
  - MM indica o mês;
  - DD indica o dia;
  - T indica o início da secção da data dedicada ao tempo;
  - hh, mm e ss indicam hora, minute e Segundo, respetivamente;
- **Exemplo:** <startdate>2002-05-30T09:00:00</startdate>

Tipos numéricos

- O tipo decimal especifica números com um máximo de 18 algarismos: `<xs:element name="prize" type="xs:decimal"/>`
- O tipo integer descreve apenas números inteiros:

```
<xs:element name="prize" type="xs:integer"/>
```

Tipos numéricos

- 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

Exemplo

- Exemplo da definição de elementos simples:

```
<xs:element name=“nome" type="xs:string"/>
<xs:element name=“nif" type="xs:integer"/>
<xs:element name="dataNascimento" type="xs:date"/>
```

Valores fixos e por defeito para elementos simples

- Um valor por defeito é associado ao elemento quando nenhum valor é especificado:

```
<xs:element name=“genero" type="xs:string" default=“masculino"/>
```

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

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

Atributos

- Se um elemento contém atributos, é considerado um elemento complexo. No entanto, o atributo é declarado como um tipo simples;
- Exemplo: `<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"/>
```

- 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"/>`

Restrições - facets

- As restrições são utilizada para restringir os valores aceitáveis para elementos e atributos;
- Restrições em elementos XML são chamadas de facets;
- Exemplo: 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>
```

Restrições - facets

- Para limitar os valores possíveis a um conjunto pré-definido, devemos utilizar enumerações:

```
<xs:element name=“automovel">
  <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>
```

Restrições - 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>
```

Restrições - 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]"/>
```

Restrições - 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])+)"/>
```

- 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}"/>`

Restrições – facets : espaço

- Podemos preservar os espaços, o que faz com que o *parser* XML não remova os espaços:

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

- Também podemos remover os espaços, enquanto que múltiplos espaços são reduzidos a um espaço:

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

Restrições – facets : restrições de tamanho

- 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"/>
```

Elementos complexos

- Um elemento complexo é um elemento XML que contém outros elementos e/ou atributos;
- Existem 4 tipos:
  - Elementos vazios;
  - Elementos que contêm outros elementos;
  - Elementos que apenas contêm texto;
  - Elementos com elementos e texto.

Elementos complexos

- Exemplo de elementos que contêm outros elementos:

```
<pessoa>
 <primeiroNome>John</primeiroNome >
 <ultimoNome>Smith</ultimoNome >
</pessoa>
```

- Duas formas de contruir 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>
```

Apenas o elemento pessoa pode utilizar este elemento complexo

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

Elementos complexos

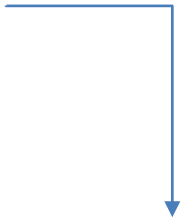
- Exemplo de um elemento que apenas contém texto e atributos:

```
<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>
```

Elementos complexos

- 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>
```

Exemplo completo XSD - Schema

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

  <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>
</xs:schema>
```

Elemento root de qualquer documento XSD

É ainda definido um prefixo para o namespace do XSD

Exemplo completo XSD - XML

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

<pessoa xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="ExemploXSD1.xsd">
   <primeiroNome>Fernando</primeiroNome>
   <ultimoNome>Santos</ultimoNome>
</pessoa>
```

Exercício 1: Acrescente a definição de um atributo “genero” de utilização obrigatória para o elemento do tipo pessoa.

Exemplo de ferramenta para validar o XML com o seu schema:
http://www.utilities-online.info/xsdvalidation/
https://www.liquid-technologies.com/online-xsd-validator

Indicadores

- Podemos controlar como os elementos são utilizados em documentos utilizando indicadores:
  - Indicadores de ordem:
    - All
    - Choice
    - Sequence
  - Indicadores de ocorrência:
    - maxOccurs
    - minOccurs
  - Indicadores de grupo:
    - Group name
    - attributeGroup name

Indicadores

- Os indicadores de ordem permitem restringir a ordem com que os elementos surgem no documento;
- 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>
```

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>
```

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

```
<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>
```

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>
```

- O indicador: **<minOccurs>** determina o número mínimo de ocorrências:
- A configuração: **maxOccurs="unbounded"** permite que o elemento ocorra infinitamente;

Indicadores

- Indicadores de grupo são utilizados para definir conjuntos de elementos;
- Deverá ser definido um dos seguintes elementos na declaração do grupo:
  - all;
  - choice;
  - sequence.
- Exemplo de um grupo em que os seus elementos devem surgir em sequência:

```
<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>
```

Indicadores

- Grupos de atributos são definidos com a declaração de attributeGroup:

```
<xs:attributeGroup name="personattrgroup">
  <xs:attribute name="firstname" type="xs:string"/>
  <xs:attribute name="lastname" type="xs:string"/>
  <xs:attribute name="birthday" type="xs:date"/>
</xs:attributeGroup>
```

Exercício

- Altere o exercício do slide 39 de forma a que permita a definição de vários elementos do tipo pessoa

Exemplo parcial XSD - schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema>
   <xs:element name="pessoas">
      <xs:complexType>
         <xs:sequence>
            < xs:element name="pessoa" type="tipoPessoa“ maxOccurs="unbounded" />
         </xs:sequence>
      </xs:complexType>
   </xs:element>

   <xs:complexType name="tipoPessoa">
      <xs:sequence>
         <xs:element name="primeiroNome" type="xs:string"/>
         <xs:element name="ultimoNome" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
(…)
</xs:schema>
```

Exemplo Completo XSD - schema

```
<?xml version="1.0" encoding="UTF-8"?>
<pessoas xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xsi:schemaLocation="ExemploXSD2.xsd">
   <pessoa>
      <primeiroNome>Pedro</primeiroNome>
      <ultimoNome>Pinto</ultimoNome>
     <genero>Masculino</genero>
   </pessoa>
   <pessoa>
      <primeiroNome>Fernando</primeiroNome>
      <ultimoNome>Santos</ultimoNome>
      <genero>Masculino</genero>
   </pessoa>
</pessoas>
```

Exercício: Altere o documento XSD de forma a que o elemento género seja incluído como um atributo do elemento pessoa

“Modularizar” os schemas

- O elemento include permite adicionar múltiplos schemas considerando o mesmo namespace:

types.XSD 

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

(…)
</xs:schema>
```

Exercício

- Implemente um *schema* XSD que permita a criação de documentos para armazenar dados de emails (exemplo do slide 8), considerando que:
  - O elemento emails pode conter um número indeterminado de elementos do tipo email;
  - Os elementos: to, from, heading e body são obrigatórios para cada email e devem surgir por esta ordem;
  - O elemento from deverá conter o caracter '@' entre o texto definido;
  - O elemento date é opcional mas a surgir deverá ser colocado imediatamente a seguir ao elemento from;
  - O atributo type é obrigatório e deverá possuir um dos seguintes valores: "Pessoal" ou "Profissional";
  - Associe tipos de dados apropriados e de acordo com o exemplo do slide 8;

Bibliografia/referências

- https://www.w3schools.com/;
- Blokdyk, G. (2018). Extensible Markup Language XML A Complete Guide. 5STARCooks.
- Eito-Brun, R. (2017). XML-based Content Management: Integration, Methodologies and Tools (1st ed.). Chandos Publishing.

Bruno Oliveira
2018/2019

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

P.PORTO

DTD – Document type definition
Introdução a XSD – XML Schema

Processamento Estruturado de Informação