SCC0661 – Multimídia e Hipermídia

Prof.: Marcelo G. Manzato
(mmanzato@icmc.usp.br)

Aula 10 – Introdução a XML Schema.

Instituto de Ciências Matemáticas e de Computação - ICMC
Sala 4-106

Conteúdo

- XML Namespaces
- XML Schema

Introdução

- DTDs: Problemas??
  - Sintaxe diferente de XML – dificulta implementação de parsers e processamento da estrutura por aplicações
  - DTD especifica vocabulário fechado e não extensível – dificulta reuso
  - DTD NÃO permite definir tipos de dados
  - DTD possui poucos tipos pré definidos
  - DTD NÃO permite especificar quantidades
  - A única vantagem do DTD é a simplicidade

Namespaces

- Conflito de nomes

HTML:

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

XML para descrição de uma mesa:

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

Namespaces

- Outro exemplo: Scalable Vector Graphics (SVG)
  - Documentos SVG podem ser embutidos em documentos HTML ou XHTML para incluir gráficos nas páginas Web
- Elementos SVG
  - title, metadata, defs, path, text, rect, circle, ellipse, line, polyline, polygon, use, image, svg g, view switch, a, altGlyphDef, script, style, symbol, marker, clipPath, mask, linearGradient, radialGradient, pattern, filter, cursor, font, animate, set, animateMotion, animateColor, animateTransform, color-profile, font-face

Namespaces

- Se elementos diferentes com nomes iguais forem adicionados em um documento, o parser não saberá como gerenciar essas ambiguidades

- Namespaces
  - Remove a ambigüidade por meio da associação de uma URI com cada aplicação XML
  - Utilização de prefixos
  - Associação do prefixo a um namespace

Namespaces

- No exemplo de table (HTML) e table (XML):

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="http://www.example.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

  - Sintaxe: `xmlns:prefix="URI"`

Namespaces

- Declaração pode ser feita também no elemento raíz:

```
<root xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="http://www.example.com/furniture">

<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</root>
```

Namespaces

- Exemplo SVG:

```
<html>
   <body>

      <h1>My first SVG</h1>

      <s:svg xmlns:s="http://www.w3.org/2000/svg" version="1.1">
         <s:title>tooltips example</s:title>
         <s:circle cx="100" cy="50" r="40" stroke="black"
              stroke-width="2" fill="red" />
      </s:svg>

   </body>
</html>
```

Namespaces

- Atenção!
  - A URI não é utilizada pelo parser para obter informação
  - O propósito é apenas fornecer um único nome ao namespace
  - Não há garantias que o documento na URI do namespace descreva a sintaxe usada; ou que o documento exista.
  - Entretanto, companhias podem disponibilizar uma página Web com a URI especificada para fornecer informações sobre o namespace
  - Se existe uma URI para uma aplicação XML, então essa URI é uma boa escolha para a definição do namespace.

- Exemplo: namespace HTML:
  - http://www.w3.org/TR/html4/

Namespaces

- Namespaces padrão
  - Pode ser especificado para evitar de usar um prefixo em todos os elementos
  - Sintaxe: `xmlns="URI"`

HTML:

```
<table xmlns="http://www.w3.org/TR/html4/">
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

XML:

```
<table xmlns="http://www.example.com/furniture">
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

Esquema XML

- Um Esquema XML (ou XML Schema):
  - define os elementos e atributos que podem aparecer no documento
  - define a hierarquia de elementos
  - define a ordem e número de elementos
  - define o conteúdo de um elemento, incluindo tipos de dados e restrições
  - define valores padrão e fixos para elementos e atributos

Esquema XML

- Características:
  - Superconjunto dos recursos disponibilizados por DTDs
  - O esquema é um documento XML
  - Elementos especificam com rigor o conteúdo e a estrutura permitidos nas instâncias dos documentos XML válidos

em Esquemas XML...

- Elementos utilizam tipos definidos pelo autor do esquema ou pré-definidos, como string, decimal, int, binary, date, time, etc.
- Número de instâncias de um elemento pode ser rigorosamente especificado com minOccurs e maxOccurs.
- Padrão de cadeias de caracteres pode ser especificado com o uso de expressões regulares

Esquema XML - vantagens

- Comunicação segura dos dados
  - Durante a comunicação de dados entre duas aplicações, é essencial que ambas as partes tenham as mesmas "expectativas" sobre o conteúdo
    - 03-11-2004, 11-03-2004, 11 de março de 2004
    - <date type="date">2004-03-11</date>
- Extensibilidade
  - Como é escrito em XML, é possível reuso, criar novos tipos de dados derivados de tipos existentes, referenciar múltiplos esquemas no mesmo documento
- Validação
  - Documento bem formado e válido

Namespaces em XML Schema

- Os elementos são associados ao Esquema XML por meio da declaração do *namespace*

  xmlns:xsd="http://www.w3.org/2001/XMLSchema“

- O prefixo xsd: (ou xs:) é usado por convenção para denotar esse espaço de nomes
- O mesmo prefixo é utilizado para os nomes pré-definidos de Esquema XML, por exemplo xsd:string
- Alternativamente, pode-se delcarar o namespace http://www.w3.org/2001/XMLSchema como namespace padrão

Declarando um Schema (po.xsd)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<schema
        xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.com/Example01"
        xmlns:po="http://www.example.com/Example01"
        elementFormDefault="qualified">

        <element name="purchaseOrder" type="po:PurchaseOrderType"/>
        ...
        <complexType  name="PurchaseOrderType">…</complexType>


...
</schema>
```

- Necessidade de um elemento raíz
  - schema
- Definição do namespace padrão: XMLSchema
  - xmlns
- Definição do target namespace do esquema
  - Indica que todos os elementos definidos pelo esquema (qualified) pertencem ao namespace/URI informados
- Prefixo para referenciar tipos e elementos de outros namespaces
  - po:PurchaseOrderType

Uma instância XML (po.xml)

```
<?xml version="1.0"?>
<purchaseOrder  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://www.example.com/Example01  po.xsd"
                xmlns="http://www.example.com/Example01"
                orderDate="2002-06-29">
        <shipTo country="US">
                <name>Alice Smith</name>
                <street>123 Maple Street</street>
                <city>Mill Valley</city>
                <state>CA</state>
                <zip>90952</zip>
        </shipTo>
        <billTo country="US">
                <name>Robert Smith</name>
                <street>8 Oak Avenue</street>
                <city>Old Town</city>
                <state>PA</state>
                <zip>95819</zip>
        </billTo>
…
</purchaseOrder>
```

Tipos de dados

- Tipos simples de dados
  - Elementos que contêm apenas conteúdo
- Tipos complexos de dados
  - Elementos com atributos e/ou subelementos
- A declaração de atributos é sempre como tipos simples de dados

Declarando um elemento tipo simples

- Elementos que podem apenas conter texto. Não pode ter subelementos e/ou atributos
- Texto pode ser de diferentes tipos: boolean, string, date, etc., ou tipos definidos pelo usuário
- Sintaxe:
  - <element name="xxx" type="yyy"/>
    - (Assumindo que o namespace para o XML Schema é o padrão)
- Exemplo:

XML:

```
<lastname>Refsnes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

XML Schema:

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

Declarando um elemento tipo simples

- Principais simpleTypes pré-definidos

simpleType	Exemplo/descrição
string	
byte	-1 ... 126
unsignedByte	0 ... 126
integer	-126789 ... 126789
int	-1 ... 126789675
long	-1...2678967543233
short	-1, 12678
decimal	-1.23, 0, 123.4, 1000.00

simpleType	Exemplo/descrição
float	Ponto flutuante 32-bits
double	Ponto flutuante 624-bits
boolean	true, false, 1, 0
time	13:20:00.000
date	1999-05-31
ID	Usado em atributos como nos DTDs

Declarando um elemento tipo simples

- Valores default e fixed para elementos simples
  - Podem ser um OU outro
- Exemplo (xsd omitido por conveniência)
  - Default:

```
<element name="color" type="date" default="1900-01-01"/>
```

  - Fixed:

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

Usando a tag simpleType

- Novos simple types podem ser derivados a partir dos já existentes.
  - Na maioria das vezes, apenas restringem-se os valores dos simple types existentes.
    - Usa-se o elemento simpleType para definir o nome do novo simple type.
      - Sintaxe: `<simpleType name="nomeTipo">…</simpleType>`
    - Usa-se o elemento restriction para indicar o tipo existente (base) e para identificar as "facetas" que contém o conjunto de valores.
      - Sintaxe: `<restriction base="tipo">…</restriction>`

Usando a tag simpleType

- Exemplo:
  - Criar um novo tipo de inteiro chamado myInteger, com valores entre 10000 e 99999 (inclusive).
    - Deve-se restringir a base do tipo inteiro (restriction base).
    - Deve-se aplicar as facetas "minInclusive" e "maxInclusive".

```
<simpleType name="myInteger">
        <restriction base="integer">
                <minInclusive value="10000"/>
                <maxInclusive value="99999"/>
        </restriction>
</simpleType>
```

Usando a tag simpleType

- No link

  http://www.w3.org/TR/xmlschema-0/#SimpleTypeFacets

  há duas tabelas com todos os simple types e as facetas aplicáveis a cada um deles.

- As tabelas apresentam um total de 12 facetas, mas as mais importantes são minInclusive, maxInclusive (usadas no exemplo anterior), minExclusive, maxExclusive, length, enumeration e pattern.

Exemplos com facetas

- Exemplo com pattern:

```
<simpleType name="SKU">
        <restriction base="string">
                <pattern value="\d{3}-[A-Z]{2}"/>
        </restriction>
</simpleType>
```

- Foi definido um simple type chamado SKU, que:
  - Tem string como restriction base.
  - A faceta pattern restringe o valor a "três dígitos seguidos de um hifen, seguido de duas letras (ASCII) maiúsculas".

Exemplos com facetas

- Exemplo com enumeration:
  - Essa faceta define um sub conjunto de valores que um simple type pode assumir.

```
<simpleType name="USState">
        <restriction base="string">
                <enumeration value="AK"/>
                <enumeration value="AL"/>
                <enumeration value="AR"/>
                <!– e assim por diante ... -->
        </restriction>
</simpleType>
```

Declaração de atributos

- Elementos simples (simpleType) não podem conter atributos
- Se um elemento tiver, ele deverá ser declarado como complexType
- Entretanto, um atributo é declarado como tipo simples

- Sintaxe:

```
<attribute name="xxx" type="yyy"/>
```

- Exemplo:

```
XML:        <lastname lang="EN">Smith</lastname>
Schema:     <attribute name="lang" type="string"/>
```

Declaração de atributos

- Um atributo pode aparecer uma ou nenhuma vez.
- O atributo use indica se um atributo é `required`, `optional` ou `prohibited`. Exemplo:

```
<attribute name="partNum" type="SKU" use="required"/>
```

- Atenção: para associar um atributo a um elemento, este elemento deve ser do tipo complexType (visto a seguir)

Declarando complexType

- Elementos que podem conter subelementos e/ou atributos

- Quatro tipos diferentes:
  - Elementos vazios
  - Elementos que contém apenas outros elementos
  - Elementos que contém apenas texto
  - Elementos que contém outros elementos e texto

Declarando complexType

- Exemplo simples de declaração de complexType

XML:

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

XML Schema:

```
<element name="employee">
  <complexType>
    <sequence>
      <element name="firstname" type="string"/>
      <element name="lastname" type="string"/>
    </sequence>
  </complexType>
</element>
```

Declarando complexType

- Outro exemplo
  - A declaração de `PurchaseOrderType` possui outros complexTypes.
    - `PurchaseOrder` irá representar a tag pai.
    - O complexType é o tipo. É necessário um elemento.

```
<element name="purchaseOrder" type="po:PurchaseOrderType"/>
```

```
<complexType name="PurchaseOrderType">
        <sequence>
                <element name="shipTo" type="po:USAddress"/>
                <element name="billTo" type="po:USAddress"/>
                <element name="comment" type="string" minOccurs="0"/>
                <element name="items" type="po:Items"/>
        </sequence>
        <attribute name="orderDate" type="date"/>
</complexType>
```

Declarando complexType

- Elementos usam a palavra-chave `element` e atributos usam `attribute`.

```
<complexType name="USAddress" >
        <sequence>
                <element name="name" type="string"/>
                <element name="street" type="string"/>
                <element name="city" type="string"/>
                <element name="state" type="string"/>
                <element name="zip" type="decimal"/>
        </sequence>
        <attribute name="country" type="string" fixed="US"/>
</complexType>
```

Exercício

- Como ficaria então uma instância XML do esquema definido?

```
<element name="purchaseOrder" type="po:PurchaseOrderType"/>
<complexType name="PurchaseOrderType">
        <sequence>
                <element name="shipTo" type="po:USAddress"/>
                <element name="billTo" type="po:USAddress"/>
                <element name="comment" type="string" minOccurs="0"/>
                <element name="items" type="po:Items"/>
        </sequence>
        <attribute name="orderDate" type="date"/>
</complexType>
<complexType name="USAddress" >
        <sequence>
                <element name="name" type="string"/>
                <element name="street" type="string"/>
                <element name="city" type="string"/>
                <element name="state" type="string"/>
                <element name="zip" type="decimal"/>
        </sequence>
        <attribute name="country" type="string" fixed="US"/>
</complexType>
```

Declarando complexType

- Repare que é possível dar qualquer nome para o complexType. USAdress foi o nome escolhido.

- Há vários tipos pré-definidos que podemos usar nas declarações (string, decimal, float, etc).
  - Vistos na seção de declaração de simpleTypes

- <sequence> significa que todos os elementos devem estar na ordem estabelecida.
  - Outras opções são all e choice, vistas a seguir.

Declarando complexType

- all especifica que os elementos filhos podem aparecer em qualquer ordem.
  - Só elementos são permitidos.
  - Mutualmente exclusivo com os elementos sequence, choice e group

```
<element name="person">
        <complexType>
                <all>
                        <element name="firstname" type="string"/>
                        <element name="lastname" type="string"/>
                </all>
        </complexType>
</element>
```

Declarando complexType

ILEGAL!!!

```
<complexType name="PurchaseOrderType">
  <sequence>
    <all>
      <element name="shipTo" type="po:USAddress"/>
      <element name="billTo" type="po:USAddress"/>
      <element name="items"  type="po:Items"/>
    </all>
    <sequence>
      <element ref="po:comment" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </sequence>
  <attribute name="orderDate" type="date"/>
</complexType>
```

Declarando complexType

- choice especifica que apenas um elemento filho pode ocorrer.

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

Sobre minOccurs e maxOccurs

- O elemento comment é opcional em PurchaseOrderType porque o valor do atributo minOccurs é 0.
- O número mínimo vezes que um elemento pode aparecer é definido pelo atributo minOccurs.
- O número máximo de vezes que um elemento pode aparecer é definido pelo atributo maxOccurs.
- O default para minOccurs e maxOccurs é 1.
- minOccurs assume normalmente os valores 0 ou 1
- maxOccurs assume normalmente 1 ou mais. Pode também ser *unbounded*.

SimpleContent e ComplexContent

- Diferentemente de simpleType, o elemento complexType não pode ter como filho os elementos restriction e/ou extension

- Deve ter como filho intermediário os elementos simpleContent ou complexContent

  - simpleContent
    - Usado para definir extensões ou restrições em um tipo complexo que contém apenas texto
    - Elemento pai: complexType

  - complexContent
    - Usado para definir extensões ou restrições em um tipo complexo que contém elementos e/ou texto
    - Elemento pai: complexType

Exemplo de simpleContent

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

```
<element name="shoesize">
 <complexType>
    <!– como permitir a adição de conteúdo e restringir para apenas números? -->
    <attribute name="country" type="string"/>
 </complexType>
</element>
```

Exemplo de simpleContent

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

```
<element name="shoesize">
 <complexType>
    <simpleContent>
       <extension base="integer">
          <attribute name="country" type="string"/>
       </extension>
    </simpleContent>
 </complexType>
</element>
```

Exemplo de complexContent

- Extensão de um tipo complexo:

```
<element name="employee" type="ex:fullpersoninfoType"/>

<complexType name="personinfoType">
  <sequence>
    <element name="firstname" type="string"/>
    <element name="lastname" type="string"/>
  </sequence>
</complexType>

<complexType name="fullpersoninfoType">
  <complexContent>
    <extension base="ex:personinfoType">
      <sequence>
        <element name="address" type="string"/>
        <element name="city" type="string"/>
        <element name="country" type="string"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Elemento vazio

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

```
<element name="product">
  <complexType>
    <attribute name="prodid" type="positiveInteger"/>
  </complexType>
</element>
```

Esquema (po.xsd) completo para po.xml (Continua…)

```
<?xml version="1.0“ encoding=“ISO-8859-1”?>
<schema  xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.com/PO1"
        xmlns:po="http://www.example.com/PO1”
        elementFormDefault="qualified">

<element name=“purchaseOrder" type="po:PurchaseOrderType"/>
<complexType name="PurchaseOrderType">
        <sequence>
                <element name="shipTo" type="po:USAddress"/>
                <element name="billTo" type="po:USAddress"/>
                <element name="items" type="po:Items"/>
        </sequence>
        <attribute name="orderDate" type="date"/>
</complexType>
```

Esquema (po.xsd) completo para po.xml (Continua...)

```
<complexType name="USAddress">
        <sequence>
                <element name="name" type="string"/>
                <element name="street" type="string"/>
                <element name="city" type="string"/>
                <element name="state" type="string"/>
                <element name="zip" type="decimal"/>
        </sequence>
        <attribute name="country" type="string" fixed="US"/>
</complexType>
```

Esquema (po.xsd) completo para po.xml (Continua...)

```
<complexType name="Items">
        <sequence>
            <element name="item" minOccurs="0" maxOccurs="unbounded">
                <complexType>
                    <sequence>
                        <element name="productName" type="string"/>
                        <element name="quantity" type="decimal"/>
                        <element name="USPrice" type="decimal"/>
                        <element name="comment" type="string" minOccurs="0"/>
                        <element name="shipDate" type="date" minOccurs="0"/>
                    </sequence>
                    <attribute name="partNum" type="po:SKU" use="required"/>
                </complexType>
            </element>
        </sequence>
</complexType>
```

Esquema (po.xsd) completo para po.xml (Fim)

```
        <simpleType name="SKU">
                <restriction base="string">
                        <pattern value="\d{3}-[A-Z]{2}"/>
                </restriction>
        </simpleType>
</schema>
```

Bibliografia

- Site da W3C (www.w3.org)
- Tutorial www.w3schools.com
- XML Schema Part 0: Primer Second Edition.
  W3C Recommendation 28 October 2004
  http://www.w3.org/TR/xmlschema-0/