Bruno Oliveira
2018/2019

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

P.PORTO

Introdução a XSD: Namespaces

Processamento Estruturado de Informação

Introdução

- Até agora aprendemos a criar documentos XML e os respetivos documentos de regras que nos permitem definir o vocabulário admissível para a construção de documentos;
- Considere que ficou encarregue de criar os documentos XSD para definir o conceito de ingrediente, enquanto que outro colega ficou responsável pela definição dos conceitos relacionados com uma Pizza;
- Considere ainda que o nome do ingrediente tem no máximo 50 caracteres e o nome da pizza 30;

Introdução

Ingrediente.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:simpleType name="tipoNome">
        <xs:restriction base="xs:string">
            <xs:maxLength value="30" />
        </xs:restriction>
    </xs:simpleType>
(…)
</xs:schema>
```

Pizza.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:simpleType name=" tipoNome ">
        <xs:restriction base="xs:string">
            <xs:maxLength value="30" />
        </xs:restriction>
    </xs:simpleType>
(…)
</xs:schema>
```

XML Namespaces

- Por exemplo, na definição do elemento pizza:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:include schemaLocation="Ingrediente.xsd"/>
   (…)
   <xs:element name="pizza">
       <xs:complexType>
          <xs:sequence>
             <xs:element name="nome" type=“tipoNome" />
          </xs:sequence>
       </xs:complexType>
    </xs:element>
</xs:schema>
```

A schema cannot contain two global components with the same name; this schema contains two occurrences of ',name'.

Como resolver?

Ingrediente.XSD

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns="http://www.aTascadaESTG.pt/Ingrediente"
        targetNamespace="http://www.aTascadaESTG.pt/Ingrediente">
(...)
```

Com a inclusão de namespaces, o elemento import deverá ser utilizado para importar documentos XSD (ao invés do elemento include)

Pizza.XSD

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns="http://www.aTascadaESTG.pt/Pizza"
        targetNamespace="http://www.aTascadaESTG.pt/Pizza"
        xmlns:c="http://www.aTascadaESTG.pt/Ingrediente"
        elementFormDefault="qualified">
    <xs:import schemaLocation="Exemplo1Ingrediente.xsd"
namespace="http://www.aTascadaESTG.pt/Ingrediente"/>
(...)
<xs:element name="pizza">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="nome" type="name" />
            <xs:element name="ingrediente">
             <xs:complexType>
               <xs:sequence>
                  <xs:element name="nome" type="c:tipoNome" />
               (...)
```

Como resolver?

Pizza.xml (excerto)

```
<?xml version="1.0" encoding="UTF-8"?>
<pizza xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://www.aTascadaESTG.pt/Pizza"
      xsi:schemaLocation="http://www.aTascadaESTG.pt/Pizza Exemplo1Pizza.xsd">
       <nome>sasd</nome>
       <ingrediente>
          <nome>sdf</nome>
       </ingrediente>
</pizza>
```

O atributo **xsi:schemaLocation** é utilizado para associar o namespace com a localização do document XSD

XML 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;
- Estes problemas podem ser resolvidos utilizando um nome como prefixo;

XML Namespaces: O atributo xmlns

- Um namespace pode ser definido com o atributo **xmlns** na tag de início de um elemento;
- A declaração de um namespace tem a seguinte declaração: xmlns:*prefix*="*URI*".
- Exemplo:

```
<root xmlns:h=http://www.w3.org/TR/html4/ xmlns:f="https://www.w3schools.com/furniture">
  <h:table>
    <h:tr>
  (…)

  <f:table>
    <f:name>African Coffee Table</f:name>
  (…)
```

Uniform Resource Identifier (URI)

- O Uniform Resource Identifier (URI) é uma string que identifica um recurso;
- O URI mais comum é o Uniform Resource Locator (URL) que identifica um domínio da internet:

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

- Os namespace por defeito (default) para um elemento permite a utilização de prefixos em todos os elementos filho;

Exemplo completo com namespaces

```
<?xml version="1.0" encoding="UTF-8"?>
<pessoas xmlns:a=http://www.estg.ipp.pt/aluno  xmlns:b=http://www.estg.ipp.pt/pessoa>
<b:pessoa>
      <b:primeiroNome>Fernando</b:primeiroNome>
      <b:ultimoNome>Santos</b:ultimoNome>
   </b:pessoa>
   <a:pessoa>
      <a:numero>87171717</a:numero>
      <a:nomeCompleto>João Pedro...</a:nomeCompleto>
   </a:pessoa>
</pessoas>
```

Exemplo XSD

```
<?xml version="1.0" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.estg.ipp.pt"
xmlns="http://www.estg.ipp.pt" elementFormDefault="qualified">
(…)
</xs:schema>
```

- O elemento <schema> pode conter alguns atributos:
  - **xmlns:xs** - indicando que os elementos e tipos de dados estão definidos no Namespace: http://www.w3.org/2001/XMLSchema e devem ter como prefixo: **xs**:
  - **targetNamespace** - indicando que os elementos definidos por este *schema* pertencem ao Namespace: www.estg.ipp.pt . Representa o namespace esperado para as instâncias independentemente dos namespaces utilizados em documentos XSD e XML;
  - **Xmlns** – Indicando que o Namespace por defeito é www.estg.ipp.pt
  - **elementFormDefault** – indicando que todos os elementos do schema devem ser **qualificados** pelo Namespace

Referenciar um Schema XML

```
<?xml version="1.0"?>
<email
xmlns="http://www.estg.ipp.pt"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.estg.ipp.pt email.xsd">
  <to>Tove@gmail.com</to>
  <from>Jani@gmail.com</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</email>
```

xmlns – Descreve o namespace por defeito, indicando que todos os elementos utilizados neste documento estão declarados no namespace;
xsi:schemaLocation – Este atributo possui dois valores: o nome do namespace a utilizar, (“Espaço”), e o caminho para o XSD utilizado nesse namespace: “http://www.estg.ipp.pt/xml/email.xsd”;

elementFormDefault

- Quando um namespace é adicionado a um documento XML, é conhecido como “source namespace”;
- Como vimos anteriormente, é possível utilizar prefixos para diferenciar namespaces;
- Quando um ou mais elementos num documento XML estão associados com um schema, o seu (source) namespace deve corresponder ao targetNamespace do documento XSD;

elementFormDefault

Ingrediente.XSD

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns="http://www.aTascadaESTG.pt/Ingrediente"
        targetNamespace="http://www.aTascadaESTG.pt/Ingrediente">
(...)
```

Pizza.XSD

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns="http://www.aTascadaESTG.pt/Pizza"
        targetNamespace="http://www.aTascadaESTG.pt/Pizza"
        xmlns:c="http://www.aTascadaESTG.pt/Ingrediente"
        elementFormDefault="qualified">
(...)
<xs:element name="pizza">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="nome" type="name" />
            <xs:element name="ingrediente">
             <xs:complexType>
               <xs:sequence>
                 <xs:element name="nome" type="c:tipoNome"
/>
                 (...)
```

elementFormDefault

- Um elemento qualified é um elemento que está associado a um namespace, utilizando para isso um prefixo;
- Podemos também definir um elemento qualified sem o prefixo (default namespace): **xmlns=”<namespace>”;**
- Apenas pode existir um namespace associado sem prefixo por documento XML;

elementFormDefault: Global vs. Local

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns="http://www.aTascadaESTG.pt/Pizza"
        targetNamespace="http://www.aTascadaESTG.pt/Pizza"
        xmlns:c="http://www.aTascadaESTG.pt/Ingrediente"
        elementFormDefault="qualified">
(...)
<xs:element name="pizza">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="nome" type="name" />
            <xs:element name="ingrediente">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="nome" type="c:name" />
                (...)
```

Global

Local

elementFormDefault

- elementFormDefault=”qualified”
  - Significa que os elementos globais e locais do schema devem ser qualificados quando utilizados no documento;
  - Ao definir o default namespace, os elementos continuam a ser qualificados mas não é necessário utilizar o prefixo

Pizza.XSD

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns="http://www.aTascadaESTG.pt/Pizza"
        targetNamespace="http://www.aTascadaESTG.pt/Pizza"
        xmlns:c="http://www.aTascadaESTG.pt/Ingrediente"
        elementFormDefault="qualified">
(...)
```

Pizza.XML

```
<pizza xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.aTascadaESTG.pt/Pizza"
    xsi:schemaLocation="http://www.aTascadaESTG.pt/Pizza
Exemplo1Pizza.xsd">
(...)
```

elementFormDefault

- elementFormDefault=”unqualified”
  - Significa que todos os elementos globais devem ser qualified e os elementos locais unqualified quando utilizados num documento XML;
  - A utilização do default namespace não é válida quando a configuração do elemento elementFormDefault é unqualified;
- Podemos também utilizar o atributeFormDefault que segue as mesmas regras descritas para os elementos (com exceção que não existem atributos locais/globais);

Recomendação

- elementFormDefault=”qualified”: Tem a desvantagem de colocar o documento demasiado “expressivo” e confuso, mas é mais claro na associação dos elementos a um namespace. A utilização de default namespace simplifica e reduz a quantidade de prefixos;
- attributeFormDefault =“unqualified”: de forma a não gerar conflitos com o default namespace;

Exercício

- Replique o exercício apresentado nos slides iniciais:
  - Defina um documento XSD para definição do vocabulário utilizado em ingredientes (nome, categoria – Molho, Carne, Vegetal, etc);
  - Defina um documento XSD para definição do vocabulário utilizado em pizzas (nome, categoria – Carne, Queijo, Vegan, etc);
  - Crie um documento XML de exemplo, representando uma pizza com vários ingredientes;
  - Considere os vocabulários independentes entre si. Por isso, deve associar os namespaces na definição de cada vocabulário;
  - Crie um menu de restaurante que pode incorporar Pizzas (considere o vocabulário definido anteriormente) e refeições genéricas (nome). As refeições possuem ingredientes (considere o vocabulário definido anteriormente).

Bibliografia/referências

- Referências Web:
  - https://www.w3schools.com/;
  - https://www.intertech.com/Blog/xml-schema-elementformdefault-and-attributeformdefault/
  - https://www.liquid-technologies.com/xml-schema-tutorial/xsd-namespaces
- Livro;
  - Anders M. and Michel S., An introduction to XML and Web Technologies, Addison-Wesley, 2006;

Bruno Oliveira
2018/2019

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

P.PORTO

Introdução a XSD: Namespaces

Processamento Estruturado de Informação