

# 7

## XML

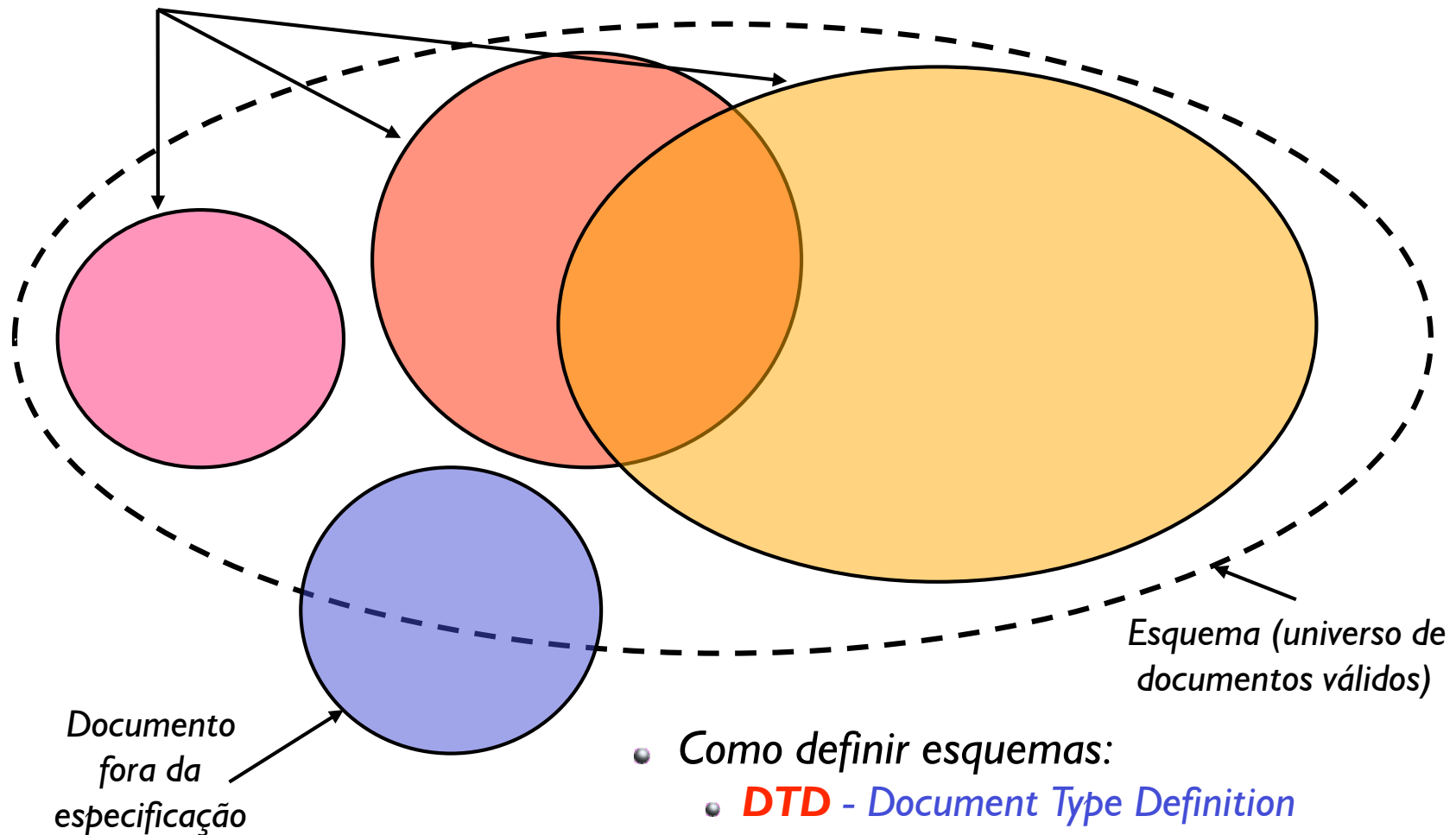
### Fundamentos de XML Schema

*Helder da Rocha*  
([helder@argonavis.com.br](mailto:helder@argonavis.com.br))

# O que é um Esquema XML?

Documentos que aderem à especificação (válidos)

- O esquema representa uma **classe**
- Os documentos são **instâncias**



- Como definir esquemas:
  - **DTD** - Document Type Definition
  - **W3C XML Schema**



# DTD vs. XML Schema

- Um esquema é essencial para que haja **comunicação usando XML**
  - *Pode ser estabelecido "informalmente" (via software)*
  - *Uso formal permite validação usando ferramentas genéricas de manipulação de XML*
- *Soluções padrão do W3C*

## DTD

```
<!ELEMENT contato  
    (nome, email, telefone)>  
<!ATTLIST contato  
    codigo NMTOKEN #REQUIRED>
```

- *Simples mas não é XML*
- *Não suporta namespaces*
- *Limitado quando a tipos de dados*

## XML Schema

```
<xsd:schema  
    xmlns:xsd=".../XMLSchema">  
<xsd:element name="contato">  
    <xsd:complexType>  
        <xsd:attribute name="codigo"  
            use="required">
```

- *É XML, porém mais complexo*
- *Suporta namespaces*
- *Permite definição de tipos*



# W3C XML Schema

- *Padrão para validação XML, modular, extensível, com amplo suporte a tipos de dados*
- **DTD**
  - *foco na estrutura*
  - *poucos recursos para controle de tipos de dados*
  - *baseado em sintaxe SGML*
  - *não suporta namespaces*
- **XML Schema**
  - *tudo são tipos de dados; grande controle sobre tipos*
  - *estruturas são tipos; tipos podem ser estendidos, criados e redefinidos*
  - *não há suporte para entidades gerais*
  - *suporte completo a namespaces*



# Exemplo: design plano - objetos globais

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

```
  <xs:element name="astro" type="astroType" />  
  <xs:element name="imagem" type="imagemType"/>
```

← Elementos

```
  <xs:attribute name="href" type="xs:anyURI" />  
  <xs:attribute name="id" type="xs:ID" />  
  <xs:attribute name="nome" type="xs:string" />  
  <xs:attribute name="diametrokm" type="xs:decimal" />
```

← Atributos

```
  <xs:complexType name="imagemType">  
    <xs:attribute ref="href" use="required" />  
  </xs:complexType>
```

← Definição de tipos de dados

```
  <xs:complexType name="astroType">  
    <xs:sequence>  
      <xs:element ref="imagem" minOccurs="0" />  
    </xs:sequence>  
    <xs:attribute ref="id" use="required" />  
    <xs:attribute ref="nome" use="required" />  
    <xs:attribute ref="diametrokm" />  
  </xs:complexType>
```

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



# Compare com um DTD

Exemplo de documento válido  
em relação a este DTD

```
<astro id="p5" nome="Jupiter">  
  <imagem href="jup31.jpg" />  
  <imagem href="jup32.jpg" />  
</astro>
```

Modelo de conteúdo  
(tipo de dados complexo)

```
<!ELEMENT astro (imagem*) >  
<!ELEMENT imagem EMPTY >
```

Elementos

Atributos

```
<!ATTLIST imagem href CDATA #REQUIRED >  
<!ATTLIST astro id ID #REQUIRED >  
<!ATTLIST astro nome CDATA #REQUIRED >  
<!ATTLIST astro diametrokm NMTOKEN #IMPLIED >
```

Atributo sempre  
associado a elemento

Tipos de dados simples  
(somente para atributos)



# Exemplo: design "boneca russa"

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="astro">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="imagem" minOccurs="0">
          <xs:complexType>
            <xs:attribute name="href" type="xs:anyURI"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="required"/>
      <xs:attribute name="nome" type="xs:string"/>
      <xs:attribute name="diametrokm" type="xs:decimal"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

*Apenas um elemento (o elemento raiz) é visível*



*Tipos não podem ser reutilizados*



*Pode haver elementos de mesmo nome em contexto diferente*

# Fundamentos XML Schema: Tipos

- Há duas qualidades de tipos
- Tipos **simples** representam tipos de dados básicos como texto, números, tokens, booleanos
  - Fazem parte do namespace do XML Schema (requerem prefixo associado ao identificador do namespace), por exemplo: *xs:int, xs:string*
- Tipos **complexos** representam estruturas do documento como entidades, atributos, etc.
  - Podem fazer parte do namespace default do próprio documento (e não necessitar de prefixo) se definidos localmente



# Fundamentos: Modelos de conteúdo

- Definem a **estrutura** de tipos **complexos**
- Modelos de conteúdo podem ser simples ou complexos
- São simples quando elemento é vazio ou quando contém apenas texto
  - Modelo de conteúdo simples pode conter atributos
- São complexos quando elemento contém outros elementos
  - Elementos podem ser definidos localmente
  - Elementos globais podem ser reutilizados



# Raiz e namespace

- *Um documento XML Schema tem a seguinte estrutura mínima*

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  ... definições de elementos, atributos, tipos
</xs:schema>
```

- *Para usá-lo, a sub-árvore a ser validada (instância) deve ter a seguinte assinatura. Exemplo:*

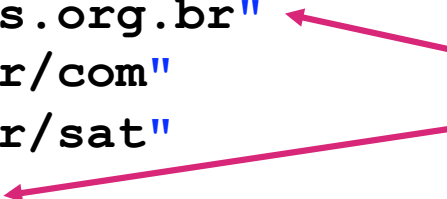
```
<sistemaEstelar
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="sistemaEstelar.xsd">
  ...
```



# Namespaces

- Schemas estimulam o uso de namespaces.
  - Os exemplos abaixo mostram uso com namespaces
- Esquema principal

```
<xs:schema
  targetNamespace="http://cosmos.org.br"
  xmlns:cm="http://cosmos.org.br/com"
  xmlns:st="http://cosmos.org.br/sat"
  xmlns="http://cosmos.org.br"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
```



- Instância

```
<se:sistemaEstelar xmlns:se="http://cosmos.org.br"
  xmlns:sat="http://cosmos.org.br/sat"
  xmlns:cmt="http://cosmos.org.br/com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://cosmos.org.br sistema.xsd
  http://cosmos.org.br/sat satelites.xsd
  http://cosmos.org.br/com cometas.xsd">
```



# Principais elementos

- É possível gerar um esquema a partir de um DTD
  - Ferramentas criam elementos e atributos
- Um esquema simples contém definições de elementos de atributos
- Elementos típicos em um esquema simples
  - `<schema>` - elemento raiz
  - `<element>`
  - `<attribute>`
  - `<simpleType>` OU `<complexType>`
  - `<simpleContent>` OU `<complexContent>`
  - `<restriction>` OU `<extension>`
  - `<enumeration>`, `<union>`, `<list>`
  - `<sequence>`, `<choice>`, `<all>`



# <element>

- *Define um elemento*
- *Deve estar associado a um tipo de dados*

```
<xs:complexType name="cometaType">  
  <xs:attribute name="id" type="xs:ID" use="required"/>  
  <xs:attribute name="nome" type="xs:string" use="required"/>  
  <xs:attribute name="planetas" type="xs:IDREFS"/>  
</xs:complexType>
```

```
<xs:element name="cometa" type="cometaType" />
```

OU

```
<xs:element name="cometa">  
  <xs:complexType>  
    <xs:attribute name="id" type="xs:ID" use="required"/>  
    <xs:attribute name="nome" type="xs:string" use="required"/>  
    <xs:attribute name="planetas" type="xs:IDREFS"/>  
  </xs:complexType>  
</xs:element>
```



## <attribute>

- *Define um atributo*
- *Pode estar embutido na definição de um tipo ou globalmente acessível (para reutilização)*

```
<xs:attribute name="raio" type="xs:decimal"/>
```

```
<xs:complexType name="sateliteType">  
  <xs:complexContent>  
    <xs:extension base="astroType">  
      <xs:attribute ref="raio" use="required"/>  
      <xs:attribute name="anoDesc" type="xs:int"/>  
    </xs:extension>  
  </xs:complexContent>  
</xs:complexType>
```



## <simpleType>

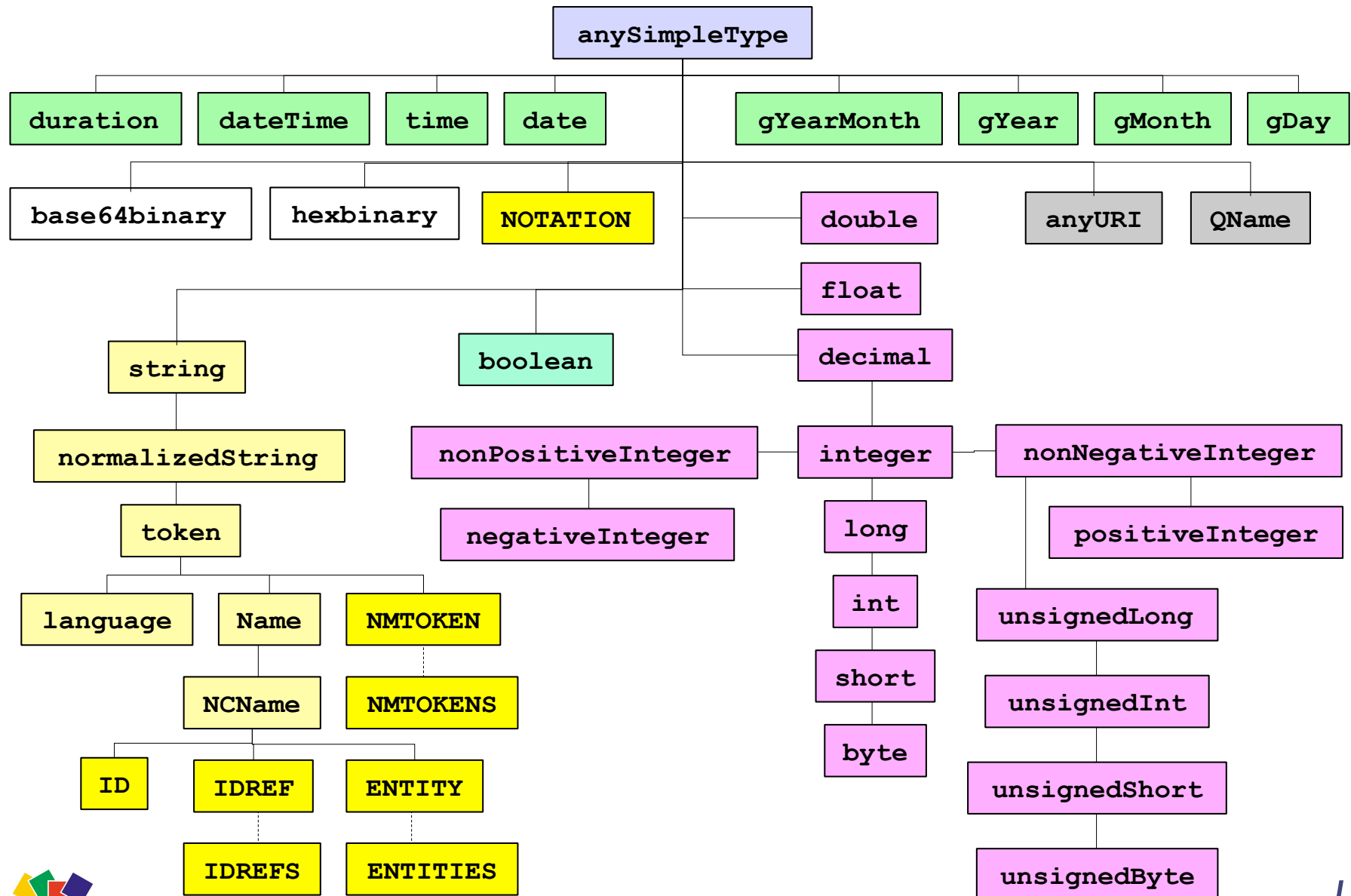
- Tipo que apenas pode conter texto
- É possível criar novos tipos a partir de derivação dos tipos existentes (globalmente acessíveis)

```
<xs:simpleType name="astroID">  
  <xs:restriction base="xs:ID">  
    <xs:pattern value="\c\d.*"/>  
  </xs:restriction>  
</xs:simpleType>
```

Expressão regular



# Tipos simples do XML Schema



# <complexType>

- *Tipo que pode conter outros elementos ou atributos*

```
<xs:complexType name="imagemType">  
  <xs:attribute name="href" type="xs:anyURI" />  
</xs:complexType>
```

```
<xs:complexType name="astroType">  
  <xs:sequence>  
    <xs:element ref="imagem" minOccurs="0" />  
    <xs:element name="satelite" type="sateliteType"  
      minOccurs="0" maxOccurs="unbounded" />  
  </xs:sequence>  
  <xs:attribute name="id" type="astroID" use="required" />  
  <xs:attribute name="nome" type="xs:token" />  
  <xs:attribute name="diametrokm" type="xs:decimal" />  
</xs:complexType>
```



## <simpleContent>

- Modelo de conteúdo simples
- Determina o tipo dos dados contido em um elemento que pode possuir atributos

```
<xs:complexType name="imagemType">  
  <xs:simpleContent>  
    <xs:restriction base="xs:string">  
      <xs:attribute name="href"  
                    type="xs:anyURI" />  
    </xs:restriction>  
  </xs:simpleContent>  
</xs:complexType>
```



## <complexContent>

- Modelo de conteúdo complexo
- Determina a organização dos elementos filho (se uma lista de opções, uma seqüência, etc).

```
<xs:complexType name="estrelaType">
  <xs:complexContent>
    <xs:extension base="astroType">
      <xs:sequence>
        <xs:element name="satelite"
          type="sateliteType"
          minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="cor" type="xs:token"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```



## <restriction> e <extension>

- *Permite restringir um tipo ou estendê-lo*
- *Podem também ser usados em modelos de conteúdo complexos para derivar tipos de outros existentes*

```
<xs:simpleType name="isbn">  
  <xs:restriction base="xs:NMTOKEN">  
    <xs:length value="10"/>  
    <xs:pattern value="[0-9]{9}[0-9X]"/>  
  </xs:restriction>  
</xs:simpleType>
```



## <sequence>

- Permite definir uma seqüência de elementos
- Equivalente ao modelo de conteúdo (a, b, c) no DTD

```
<xs:element name="sistemaEstelar">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="centro" type="centroType"/>
      <xs:element name="orbita" type="orbitaType"
        minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="cometa" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Equivalente, em DTD, a (centro, orbita\*, cometa\*)



## <choice>

- Permite escolher um elemento de um conjunto
  - Ou mais, caso *maxOccurs* seja "unbounded"
- Equivalente ao modelo de conteúdo  $(a \mid b \mid c)$  no DTD

```
<xs:complexType name="orbitaType">
  <xs:choice>
    <xs:element name="estrela" type="estrelaType"/>
    <xs:element name="planeta" type="sateliteType"/>
    <xs:element name="asteroide" type="sateliteType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:choice>
  <xs:attribute name="raioMedUA" type="xs:decimal"/>
</xs:complexType>
```

Equivalente a  $(\text{estrela} \mid \text{planeta} \mid \text{asteroide}^*)$



## <import>

- *Permite importar sub-esquemas*
- *É preciso definir o namespace usando xmlns no elemento raiz*
- *Namespace deve coincidir com namespace definido nos sub-schemas*

```
<xs:import  
  namespace="http://www.cosmos.org.br/satelites"  
  schemaLocation="satelites.xsd" />
```

```
<xs:import  
  namespace="http://www.cosmos.org.br/cometas"  
  schemaLocation="cometas.xsd" />
```

