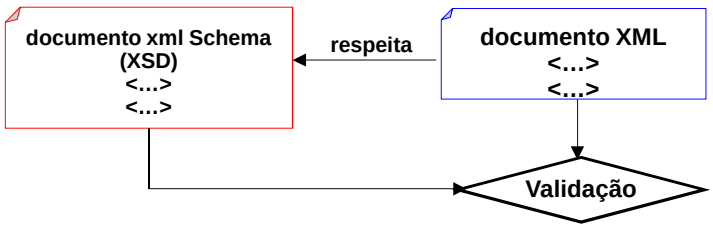
• *XML Schema*

(Revisão de conceitos fundamentais)

XML SCHEMA

- Define uma *Markup Language* para documentos XML
- Um *schema* descreve a estrutura de um documento XML em termos de tipos (*complex types* e *simple types*).
  - *complex types* – descrevem como os elementos XML estão organizados e encaixam (*nested*);
  - *simple types* – tipos de dados primitivos (int, string, …) contidos nos elementos XML e atributos;
- Um *parser* de documentos XML, que suporte XML *schemas,* pode validar o conteúdo do documento XML através da definição do *schema* XML a que obedece o documento;



XML SCHEMA – Define uma <PTAddress> *Markup Language*

definição de um prefixo (myns) para o targetNamespace

```
<?xml version="1.0" encoding="utf-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:myns="http://SD"
        targetNamespace="http://SD"
        elementFormDefault="qualified" >
    <element name="address" type="myns:PTAddress" />
    <complexType name="PTAddress">
      <sequence>
        <element name="nome" type="string" />
        <element name="rua"  type="string" />
        <element name="numero" type="int" minOccurs="1" maxOccurs="3" />
        <element name="cidade" type="string" />
        <element name="codigo-postal" type="string" />
      </sequence>
    </complexType>
</schema>
```

targetNamespace - XML *namespace* para todos os tipos criados explicitamente neste *schema*

Instância da <PTAddress> *Markup Language*

```
<?xml version="1.0" encoding="utf-8" ?>
<addr:address xmlns:addr="http://SD">
   <addr:nome>ISEL</addr:nome>
   <addr:rua>Conselheiro Emídio Navarro</addr:rua>
   <addr:numero>1</addr:numero>
   <addr:numero>2</addr:numero>
   <addr:cidade>Lisboa</addr:cidade>
   <addr:codigo-postal>1950-062</addr:codigo-postal>
</addr:address>
```

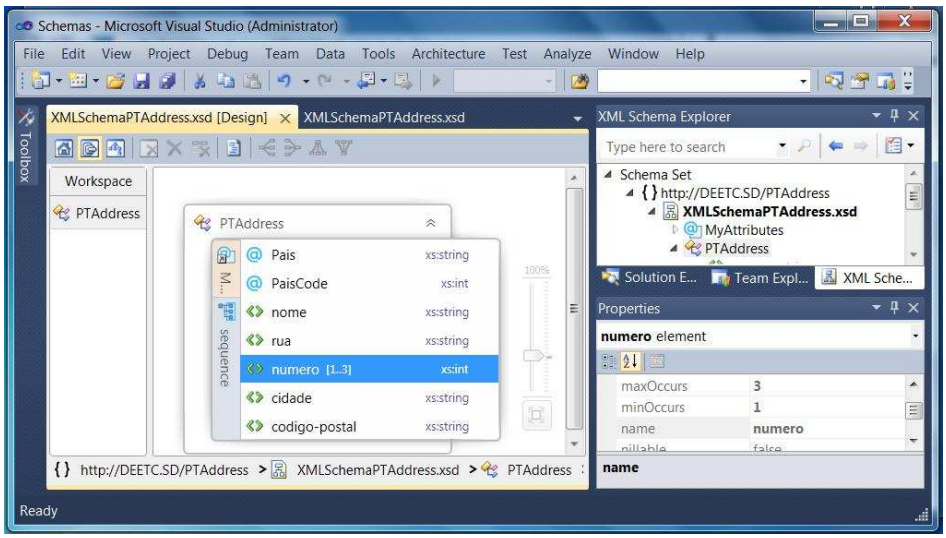
Note que se usam Qualified Names (<addr:address>, <addr:nome> etc.) porque no *Schema* se definiu o atributo elementFormDefault="qualified"

Utilização do atributo *schemaLocation* com documentos XML

```
<?xml version="1.0" encoding="utf-8" ?>
<addr:address xmlns:addr="http://SD"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://SD
                          http://www.isel.pt/sd/PTAddress.xsd">
    <addr:nome>ISEL</addr:nome>
    <addr:rua>Conselheiro Emídio Navarro</addr:rua>
    <addr:numero>1</addr:numero>
    <addr:numero>2</addr:numero>
    <addr:cidade>Lisboa</addr:cidade>
    <addr:codigo-postal>1959-007</addr:codigo-postal>
</addr:address>
```

pares formados por *namespace, Location* (*Location* é o URL onde está localizado o *schema*)

XML schema Designer no Visual Studio 2010



XML schema com atributos

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="XMLSchemaPTAddress“
          targetNamespace=“http://SD/PTAddress”
          xmlns="http://SD/PTAddress"
          xmlns:myns="http://SD/PTAddress"
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          attributeFormDefault="qualified" elementFormDefault="qualified">

 <xs:attributeGroup name="MyAttributes"  >
     <xs:attribute name="Pais" type="xs:string" use="required" form="qualified"  />
     <xs:attribute name="PaisCode" type="xs:int" use="required" form="qualified"  />
 </xs:attributeGroup>

 <xs:element name="address" type="PTAddress">  </xs:element>

 <xs:complexType name="PTAddress">
     <xs:sequence>
        <xs:element name="nome" type="xs:string" />
        <xs:element name="rua" type="xs:string" />
        <xs:element name="numero" type="xs:int" minOccurs="1" maxOccurs="3" />
        <xs:element name="cidade" type="xs:string" />
        <xs:element name="codigo-postal" type="xs:string" />
     </xs:sequence>
     <xs:attributeGroup ref="MyAttributes"   />
 </xs:complexType>

</xs:schema>
```

Um *PTAddress* tem de ter dois atributos: Pais e PaisCode

Validar fragmento XML face ao *schema XMLSchemaPTAddress*

```
using System;
using System.Xml;
using System.Xml.Schema;
namespace ValidarXML {

   class Program {
      static erro=false;

      public static void ShowParserErrors(object sender, ValidationEventArgs args) {
         Console.WriteLine("ERRO: XML fragment não obedece ao schema: {0}", args.Message);
          erro=true;
      }

      static void Main(string[] args) {
      try {                                   //XML fragment para validar
          string xmlFrag =
             "<address myns:Pais='port' myns:PaisCode='351' xmlns='http://SD/PTAddress.xsd'
                                          xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'  >" +
               "<nome>ISEL</nome>" +
               "<rua>Conselheiro Emidio Navarro</rua>" +
               "<numero>1</numero>" +
               "<numero>2</numero>" +
               "<numero>3</numero>" +
               //"<numero>4</numero>" +
               "<cidade>Lisboa</cidade>" +
               "<codigo-postal>1950</codigo-postal>" +
             "</address>";
. . .
```

Exemplo: XMLschema.zip

Validar fragmento XML face ao *schema XMLSchemaPTAddress (cont.)*

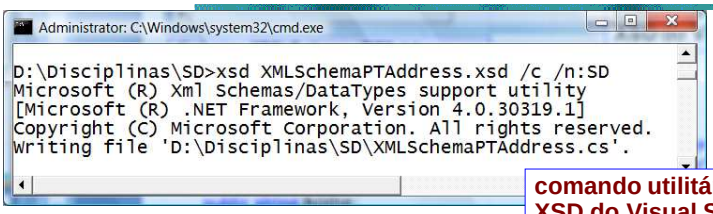
```
 . . .

  //Set the settings to xml reader and add the schema
  XmlReaderSettings settings = new XmlReaderSettings();
  settings.Schemas.Add("http://SD/XMLSchemaPTAddress.xsd", " ..\\..\\XMLSchemaPTAddress.xsd");
  settings.ValidationType = ValidationType.Schema;
  settings.ValidationFlags |= XmlSchemaValidationFlags.AllowXmlAttributes;
 settings.ValidationFlags |= XmlSchemaValidationFlags.ReportValidationWarnings;
 settings.ConformanceLevel = ConformanceLevel.Fragment;
 settings.ValidationEventHandler += new ValidationEventHandler(ShowParserErrors);
 // Create the XmlNamespaceManager.
 XmlNamespaceManager nsmgr = new XmlNamespaceManager(new NameTable());
 nsmgr.AddNamespace("myns", "http://SD/PTAddress.xsd");
 // Create the XmlParserContext.
 XmlParserContext context = new XmlParserContext(null, nsmgr, null, XmlSpace.None);
 XmlReader reader = XmlReader.Create(new StringReader(xmlFrag), settings, context);

  while (reader.Read()) {     }  // ler toda a string com fragmento xml
  if (!erro) Console.WriteLine("XML fragment VÁLIDO");
}
catch (XmlException XmlExp) {  Console.WriteLine(XmlExp.Message);    }
catch(XmlSchemaException XmlSchExp) {   Console.WriteLine(XmlSchExp.Message);   }
catch(Exception GenExp) {     Console.WriteLine(GenExp.Message);   }
finally { Console.Read();  }
}//main

}//class
}//namespace
```


Classe C# gerada a partir do *XML schema* <PTAddress>



comando utilitário XSD do Visual Studio .NET

ficheiro XMLSchemaPTAddress.cs

```
// This source code was auto-generated by xsd, Version=4.0.30319.1.
namespace SD {
   using System.Xml.Serialization;

   [System.Xml.Serialization.XmlTypeAttribute(Namespace="http://SD/PTAddress.xsd")]
   [System.Xml.Serialization.XmlRootAttribute("address", Namespace="http://SD/PTAddress.xsd . . . )]
   public partial class PTAddress {
      public string nome;
      public string rua;
      [System.Xml.Serialization.XmlElementAttribute("numero")]
      public int[] numero;
      public string cidade;
      [System.Xml.Serialization.XmlElementAttribute("codigo-postal")]
      public string codigopostal;
   }
}
```

XML Namespaces importantes

- xmlns="http://www.w3.org/2001/XMLSchema"
  - *Namespace* standard definido pelo W3C para especificação do *XML schema;*
  - Define todos os elementos e atributos que podem ser usados num XML schema (element, sequence, complexType, targetNamespace, minOccurs, maxOccurs, name, type etc.);
  - Define um conjunto de tipos simples (int, float, string, boolean, time, date, dateTime, decimal, unsigned short etc.)
- xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  - *Namespace* standard definido pelo W3C para especificação de alguns atributos que podem ser usados em documentos XML, por exemplo xsi:schemaLocation, xsi:type