Inferring Schemas from XML Documents

.NET Framework (current version)

This topic describes how to use the XmlSchemaInference class to infer an XML Schema definition language (XSD) schema from the structure of an XML document.

The Schema Inference Process

The XmlSchemaInference class of the System.Xml.Schema namespace is used to generate one or more XML Schema definition language (XSD) schemas from the structure of an XML document. The generated schemas may be used to validate the original XML document.

As an XML document is processed by the XmlSchemaInference class, the XmlSchemaInference class makes assumptions about the schema components that describe the elements and attributes in the XML document. The XmlSchemaInference class also infers schema components in a constrained way by inferring the most restrictive type for a particular element or attribute. As more information about the XML document is gathered, these constraints are loosened by inferring less restrictive types. The least restrictive type that can be inferred is `xs:string`.

Take, for example, the following piece of an XML document.

```
<parent attribute1="6">
    <child>One</child>
    <child>Two</child>
</parent>
<parent attribute1="A">
```

In the example above, when the `attribute1` attribute is encountered with a value of 6 by the XmlSchemaInference process, it is assumed to be of type `xs:unsignedByte`. When the second `parent` element is encountered by the XmlSchemaInference process, the constraint is loosened by modifying the type to `xs:string` because the value of the `attribute1` attribute is now A. Similarly, the `minOccurs` attribute for all the `child` elements inferred in the schema are loosened to `minOccurs="0"` because the second parent element has no child elements.

Inferring Schemas from XML Documents

The XmlSchemaInference class uses two overloaded InferSchema methods to infer a schema from an XML document.

The first XmlSchemaInference.InferSchema method is used to create a schema based on an XML document. The second XmlSchemaInference.InferSchema method is used to infer a schema that describes multiple XML documents. For example, you can feed multiple XML documents to the XmlSchemaInference.InferSchema method one at a time to produce a schema that describes the entire set of XML documents.

The first XmlSchemaInference.InferSchema method infers a schema from an XML document contained in an XmlReader object, and returns an XmlSchemaSet object containing the inferred schema. The second XmlSchemaInference.InferSchema method searches an XmlSchemaSet object for a schema with the same target namespace as the XML document contained in the XmlReader object, refines the existing schema, and returns an XmlSchemaSet object containing the inferred schema.

The changes made to the refined schema are based on new structure found in the XML document. For example, as an XML document is traversed, assumptions are made about the data types found, and the schema is created based on those assumptions. However, if data is encountered on a second inference pass that differs from the original assumption, the schema is refined. The following example illustrates the refinement process.

C#

```
        XmlReader reader = XmlReader.Create("item1.xml");
        XmlReader reader1 = XmlReader.Create("item2.xml");
        XmlSchemaSet schemaSet = new XmlSchemaSet();
        XmlSchemaInference inference = new XmlSchemaInference();
        schemaSet = inference.InferSchema(reader);

        // Display the inferred schema.
        Console.WriteLine("Original schema:\n");
        foreach (XmlSchema schema in
schemaSet.Schemas("http://www.contoso.com/items"))
        {
            schema.Write(Console.Out);
        }

        // Use the additional data in item2.xml to refine the original schema.
        schemaSet = inference.InferSchema(reader1, schemaSet);

        // Display the refined schema.
        Console.WriteLine("\n\nRefined schema:\n");
        foreach (XmlSchema schema in
schemaSet.Schemas("http://www.contoso.com/items"))
        {
            schema.Write(Console.Out);
        }
```

The example takes the following file, `item1.xml`, as its first input.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<item xmlns="http://www.contoso.com/items" productID="123456789">
        <name>Hammer</name>
        <price>9.95</price>
        <supplierID>1929</supplierID>
</item>
```

The example then takes the `item2.xml` file as its second input:

XML

```
<?xml version="1.0" encoding="utf-8"?>
<item xmlns="http://www.contoso.com/items" productID="A53-246">
        <name>Paint</name>
        <price>12.50</price>
</item>
```

When the `productID` attribute is encountered in the first XML document, the value of `123456789` is assumed to be an `xs:unsignedInt` type. However, when the second XML document is read and the value of `A53-246` is found, the `xs:unsignedInt` type can no longer be assumed. The schema is refined and the type of `productID` is changed to `xs:string`. In addition, the `minOccurs` attribute for the `supplierID` element is set to `0`, because the second XML document contains no `supplierID` element.

The following is the schema inferred from the first XML document.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://www.contoso.com/items"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="item">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="price" type="xs:decimal" />
        <xs:element name="supplierID" type="xs:unsignedShort" />
      </xs:sequence>
      <xs:attribute name="productID" type="xs:unsignedInt" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The following is the schema inferred from the first XML document, refined by the second XML document.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://www.contoso.com/items"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="item">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="price" type="xs:decimal" />
        <xs:element minOccurs="0" name="supplierID" type="xs:unsignedShort" />
      </xs:sequence>
      <xs:attribute name="productID" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Inline Schemas

If an inline XML Schema definition language (XSD) schema is encountered during the XmlSchemaInference process, an XmlSchemaInferenceException is thrown. For example, the following inline schema throws an XmlSchemaInferenceException.

```
<root xmlns:ex="http://www.contoso.com" xmlns="http://www.tempuri.org">
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.contoso.com">
        <xs:element name="Contoso" type="xs:normalizedString" />
    </xs:schema>
    <ex:Contoso>Test</ex:Contoso>
</root>
```

Schemas that Cannot be Refined

There are W3C XML Schema constructs that the XML Schema definition language (XSD) schema XmlSchemaInference process cannot handle if given a type to refine and cause an exception to be thrown. Such as a complex type whose top-level compositor is anything other than a sequence. In the Schema Object Model (SOM), this corresponds to an XmlSchemaComplexType whose Particle property is not an instance of XmlSchemaSequence.

See Also

XmlSchemaInference
XML Schema Object Model (SOM)
Inferring an XML Schema
Rules for Inferring Schema Node Types and Structure
Rules for Inferring Simple Types