Tip: When to use local and global declarations

Two types of elements in XML Schema

Benoit Marchal

September 11, 2003

W3C XML Schema offers many powerful options for structuring and organizing your XML vocabularies. In this tip, Benoit compares global and local declarations of elements, and provides pointers on when to use which.

View more content in this series

W3C XML Schema is becoming increasingly popular because XML developers need to overcome the limitations of the DTD as a modeling language. XML Schema is particularly popular because it offers a richer set of data types. With XML Schema, it is easy to specify whether an element should contain text, a number, a boolean, or some other data type.

Still, XML Schema offers more than just data typing. It has options for organizing element declarations to improve readability and maintainability. In this tip, I'll show you the scoping option available in XML Schema.

Local versus global declarations

XML Schema allows you to declare elements immediately under the `xs:schema` element or as part of another declaration. Declarations that appear directly under the `xs:schema` element have a **global** scope while other declarations have a **local** scope. Global declarations are not new -- they were originally introduced with DTDs -- but local declarations are unique to XML Schema.

The most noticeable difference between global and local declarations of elements is that global declarations can be referenced from other declarations, whereas local declarations only exist within their local scope. A practical side effect is that global declarations must be unique: No two global declarations can use the same element name. Local declarations, on the other hand, cause no conflict if they have different contexts. It is even possible to override a global declaration in a local one.

Why would you want to override declarations? This is mostly useful when designing large schemas and, more specifically, new schemas that import declarations from existing schemas. Indeed, in the latter case, the imported schemas might have conflicting declarations.
For example, one schema could define a key element as base64 data for storing a cryptographic key while another schema would define key as an integer for storing a database identifier. Here, the key element has two different definitions, which will cause a problem if you try to create a third schema that imports the first two.

**Looking at examples**

The schema in Listing 1 uses global declarations exclusively. The `<xs:element name="...">` construct defines a new element; it is similar to the `<!ELEMENT ...>` statement in a DTD. These declarations may reference other declarations through the `<xs:element ref="..."/>` construct.

**Listing 1. Global declarations**

```xml
<?xml version="1.0"?>
<xs:schema targetNamespace="http://ananas.org/2003/tips/local"
    xmlns:tp="http://ananas.org/2003/tips/local"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="list">
    <xs:complexType>
    <xs:sequence>
    <xs:element ref="tp:movie" maxOccurs="unbounded"/>
    </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="movie">
    <xs:complexType>
    <xs:sequence>
    <xs:element ref="tp:title"/>
    <xs:element ref="tp:genre" maxOccurs="unbounded"/>
    </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="title" type="xs:string"/>

<xs:element name="genre" type="xs:string"/>
</xs:schema>
```

Listing 2, on the other hand, uses local declarations as much as possible. The only global declaration is for the root of the document (the root of a document has to be a global declaration, since the root has no parent). All the other declarations are nested under the root declaration. The extensive nesting has earned this approach the nickname "Russian dolls."

**Listing 2. Local declarations**

```xml
<?xml version="1.0"?>
<xs:schema targetNamespace="http://ananas.org/2003/tips/local"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">

<xs:element name="list">
    <xs:complexType>
    <xs:sequence>
    <xs:element name="movie" maxOccurs="unbounded">
    <xs:complexType>
    <xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="title" type="xs:string"/>

<xs:element name="genre" type="xs:string"/>
</xs:schema>
```
References and namespaces

When referencing a global element definition, you need to provide a fully qualified name, including the appropriate namespace prefix (such as ref="tp:title"). However, the element declaration is a non-qualified name (name="title"). The (somewhat twisted) logic behind this behavior is: Every declaration in the schema automatically falls into the target namespace. Yes, it's somewhat confusing, but at least you've been warned.

Note the use of the elementFormDefault attribute. By default, elements with a local declaration are unqualified (they take no namespace prefix) in the XML document. The reasoning behind this is that unqualified elements are similar to attributes, which are unqualified by default.

Listing 3 illustrates unqualified local elements. Here, the movie, title, and genre elements have no namespace prefix; it is assumed that they are in the same namespace as their parent. The jury is still out on the use of unqualified local elements. SOAP has had some success using them, but most users find them confusing.

A word of warning: You would need to change the elementFormDefault in Listing 2 from qualified to unqualified in order to validate Listing 3.

Listing 3. Unqualified local elements

```xml
<?xml version="1.0"?>
  <movie>
    <title>Johnny English</title>
    <genre>comedy</genre>
  </movie>
  <movie>
    <title>Finding Nemo</title>
    <genre>family</genre>
  </movie>
</tp:list>
```

Obviously, if you want unqualified elements in your documents then you have to use local elements.

Drawing conclusions

So which one is best -- local or global declarations? Alas, as is often the case with design strategies, the answer is not black or white, and most schemas use both. Local declarations are interesting because they isolate a portion of the schema. The local names are opaque to other contexts, which minimizes the risks of name conflicts.
Another benefit of local declarations is that they clearly identify the possible roots in a document.

The main drawback of local declarations is that they prevent reuse. With Listing 2 (local declarations), it is impossible to reuse the movie element in another context. That would have been easy with Listing 1 (global declarations).

In conclusion, the best choice depends on the nature of the element: A self-contained element that you expect to reuse elsewhere in the schema is a prime candidate for global declaration; elements that only make sense in a given context should be declared locally.

Assuming that movie is a self-contained element, then Listing 4 is a better schema with a good mix of local and global declarations.

**Listing 4. Mixing declarations**

```xml
<?xml version="1.0"?>
<xs:schema targetNamespace="http://ananas.org/2003/tips/local"
            xmlns:tp="http://ananas.org/2003/tips/local"
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified">
  <xs:element name="list">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tp:movie" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="movie">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="genre" type="xs:string"
                    maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Although not covered in this tip, you can also use type definitions in association with global and local element declarations to improve the readability of your schemas.
Related topics

- Find out all about XML Schema at the W3C site.
- Get the basics of using XML Schema to define elements with this introduction to writing XML schemas by Ashvin Radiya and Vibha Dixit (developerWorks, August 2000).
- Visit xFront's Best Practices page for pointers on using global and local elements.
- One of the best references on the topic is XML Schema by Eric Van Der Vlist (O'Reilly & Associates, 2002).
- Find more XML resources on the developerWorks XML zone. For a complete list of XML tips to date, check out the tips summary page.
- IBM's DB2 (now known as Information Management) database provides not only relational database storage. Visit the Information Management content area to learn more about DB2.
- IBM trial software: Build your next development project with trial software available for download directly from developerWorks.
- Find out how you can become an IBM Certified Developer.

© Copyright IBM Corporation 2003
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)