

Global versus Local

[aus <http://www.xfront.com/BestPracticesHomepage.html>]

Issue

When should an element or type be declared global versus when should it be declared local?

Recall that a component (element, complexType, or simpleType) is "global" if it is an immediate child of <schema>, whereas it is "local" if it is not an immediate child of <schema>, i.e., it is nested within another component.

Russian Doll Design

This design approach has the schema structure mirror the instance document structure, e.g., declare a Book element and within it declare a Title element followed by an Author element:

```
<xsd:element name="Book">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="Author" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</element>
```

The instance document has all its components bundled together. Likewise, the schema is designed to bundle together all its element declarations.

Salami Slice Design

With this design we disassemble the instance document into its individual components. In the schema we define each component (as an element declaration), and then assemble them together:

```
<xsd:element name="Title" type="xsd:string"/>
```

```
<xsd:element name="Author" type="xsd:string"/>
```

```
<xsd:element name="Book">  
  <xsd:complexType>  
    <xsd:sequence>  
      <xsd:element ref="Title"/>  
      <xsd:element ref="Author"/>  
    </xsd:sequence>  
  </xsd:complexType>  
</xsd:element>
```

Venetian Blind Design

With this design approach we disassemble the problem into individual components, as the Salami Slice design does, but instead of creating element declarations, we create type definitions. Here's what our example looks like with this design approach:

```
<xsd:simpleType name="Title">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Mr."/>
    <xsd:enumeration value="Mrs."/>
    <xsd:enumeration value="Dr."/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="Name">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="Publication">
  <xsd:sequence>
    <xsd:element name="Title" type="Title"/>
    <xsd:element name="Author" type="Name"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="Book" type="Publication"/>
```

Best Practice

1. The Venetian Blind design is the one to choose where your schemas require the flexibility to turn namespace exposure on or off with a simple switch, and where component reuse is important.
2. Where your task requires that you make available to instance document authors the option to use element substitution, then use the Salami Slice design.
3. Where minimizing size and coupling of components is of utmost concern then use the Russian Doll design.