XML Schema Design Patterns

Sang Shin
Michèle Garoche
www.javapassion.com
"Learning is fun!"

Agenda

- Popular XML Schema Design Patterns
  - Russian Doll
  - Salami Slice
  - Venetian Blind
  - Garden of Eden
- NetBeans Support on XML Schema Design Patterns

Popular XML Schema Design Patterns

Popular XML Schema Design Patterns

- Russian Doll
- Salami Slice
- Venetian Blind
- Garden of Eden

Popular XML Schema Design Patterns

- The patterns vary according to the number of their global elements or types.
- You can conveniently classify the four most common patterns according to two criteria, namely:
  - Ease of use for instance developers
  - Ease of reuse for schema developers

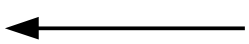
Russian Doll

Russian Doll

- The Russian Doll design contains only one single global element.
- All the other elements are local.
- You nest element declarations within a single global declaration

Example: Russian Doll

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://schemas.sun.com/point/russiandoll"
        xmlns:tns="http://schemas.sun.com/point/russiandoll"
        elementFormDefault="qualified">

  <xsd:element name="Line">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="PointA">
          <xsd:complexType>
            <xsd:attribute name="x" type="xsd:integer"/>
            <xsd:attribute name="y" type="xsd:integer"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="PointB">
          <xsd:complexType>
            <xsd:attribute name="x" type="xsd:integer"/>
            <xsd:attribute name="y" type="xsd:integer"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```



Global element

Advantages & Disadvantages

- Advantages
  - Contains only one valid root element.
  - Could reduce the complexity of namespace
  - Easy to use from instance developer's point of view
- Disadvantages
  - Reuse of elements is limited – not easy to use from schema developer's point of view
  - Supports single-file schema only

Usage

- Russian Doll is the simplest and easiest pattern to use by instance developers.
- However, if its elements or types are intended for reuse, Russian Doll is not suitable for schema developers.

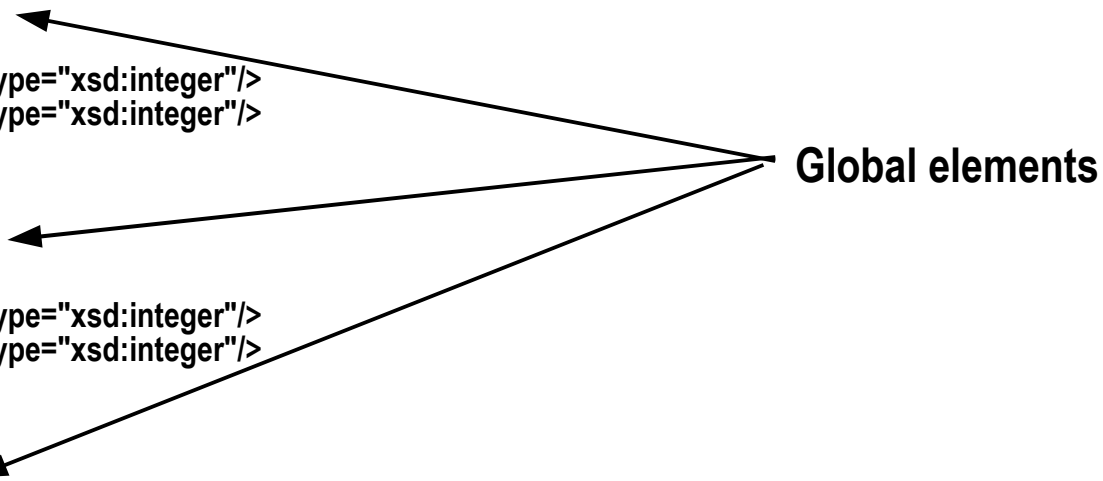
Salami Slice

Salami Slice

- All the elements in the Salami Slice design are global.
- No nesting of element declarations is required and you can reuse the declarations throughout the schema.
- You must define all the elements within the global namespace.

Example: Salami Slice

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://schemas.sun.com/point/salami"
        xmlns:tns="http://schemas.sun.com/point/salami"
        xmlns="http://schemas.sun.com/point/salami"
        elementFormDefault="qualified">

   <xsd:element name="PointA">
      <xsd:complexType>
         <xsd:attribute name="x" type="xsd:integer"/>
         <xsd:attribute name="y" type="xsd:integer"/>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="PointB">
      <xsd:complexType>
         <xsd:attribute name="x" type="xsd:integer"/>
         <xsd:attribute name="y" type="xsd:integer"/>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="Line">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="PointA"/>
            <xsd:element ref="PointB"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```



Advantages & Disadvantages

- Advantages
  - Contains all reusable elements (from schema designer's standpoint)
  - Supports reuse of elements from other documents.
- Disadvantages
  - Exposes the complexity in namespace.

Venetian Blind

Venetian Blind

- The Venetian Blind design contains only one global element.
- All the other elements are local.
- You nest element declarations within a single global declaration by means of named complex types and element groups.
- You can reuse those types and groups throughout the schema and must define only the root element within the global namespace.

Example: Venetian Blind

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://schemas.sun.com/point/venetianblind"
        xmlns:tns="http://schemas.sun.com/point/venetianblind"
        xmlns="http://schemas.sun.com/point/venetianblind"
        elementFormDefault="qualified">

   <xsd:complexType name="PointType">
      <xsd:attribute name="x" type="xsd:integer"/>
      <xsd:attribute name="y" type="xsd:integer"/>
   </xsd:complexType>

   <xsd:element name="Line">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="PointA" type="PointType"/>
            <xsd:element name="PointB" type="PointType"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Advantages & Disadvantages

- Advantages
  - Contains only one single root element.
  - Allows reuse for all the types and the single global element.
  - Allows multiple files
- Disadvantages
  - Limits encapsulation by exposing types.

Usage

- Because it has only one single root element and all its types are reusable, Venetian Blind is suitable for use by both instance developers and schema developers.
  - Venetian Blind is considered as an extension of Russian Doll, in which all the types are defined globally.

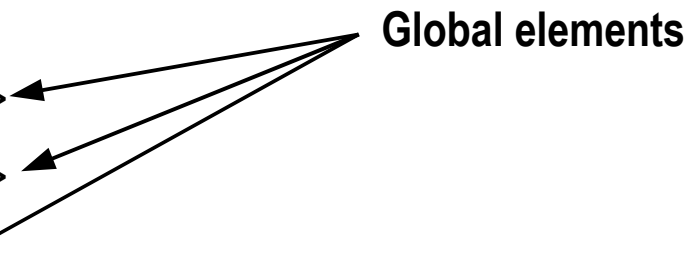
Garden of Eden

Garden of Eden

- You define all the elements and types in the global namespace and refer to the elements as required.

Example: Garden of Eden

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://schemas.sun.com/point/gardenofeden"
        xmlns="http://schemas.sun.com/point/gardenofeden"
        elementFormDefault="qualified">

   <xsd:complexType name="PointType">
      <xsd:attribute name="x" type="xsd:integer"/>
      <xsd:attribute name="y" type="xsd:integer"/>
   </xsd:complexType>

   <xsd:complexType name="LineType">
      <xsd:sequence>
         <xsd:element ref="PointA"/>
         <xsd:element ref="PointB"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:element name="PointA" type="PointType"/>

   <xsd:element name="PointB" type="PointType"/>

   <xsd:element name="Line" type="LineType"/>
</xsd:schema>
```



Advantages & Disadvantages

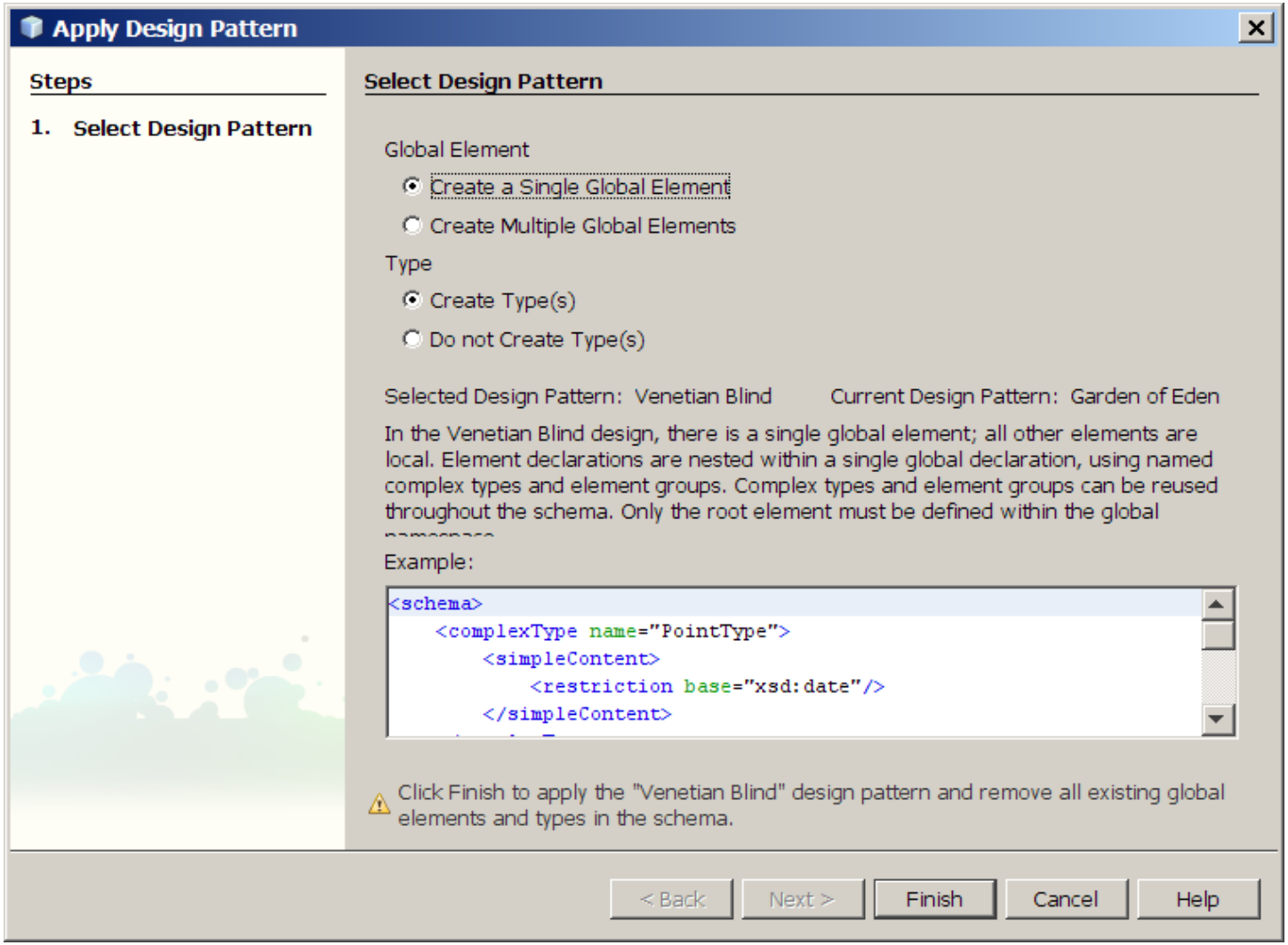
- Advantages
  - Allows reuse of both elements and types.
  - Allows multiple files.
- Disadvantages
  - Contains many potential root elements.
  - Limits encapsulation by exposing types.
  - Is difficult to read and understand.

NetBeans Support on XML Schema Design Patterns

NetBeans XML Schema Design Support

- Lets you choose a design pattern by letting you choose
  - Scheme of Creating global element
  - Scheme of Creating type
- Creating global element
  - Create a single global element
  - Create Multiple global elements
- Creating type
  - Create Type(s)
  - Do not Create Type(s)

NetBeans XML Schema Design Support



Thank you!

Sang Shin
Michèle Garoche
http://www.javapassion.com
“Learning is fun!”