XML Schemas and Namespaces

Adapted from a tutorial by Roger L. Costello.

What are XML Schemas?

- Answer: An XML vocabulary for expressing your data's business rules (or constraints)

Example

```
<location>
        <latitude>32.904237</latitude>
        <longitude>73.620290</longitude>
        <uncertainty units="meters">2</uncertainty>
</location>
```

Is this data valid?

To be valid, it must meet these **constraints (data business rules)**:

1. The location must be comprised of a latitude, followed by a longitude, followed by an indication of the uncertainty of the latitude/longitude measurements.
2. The latitude must be a decimal with a value between -90 to +90
3. The longitude must be a decimal with a value between -180 to +180
4. For both latitude and longitude the number of digits to the right of the decimal point must be exactly six digits.
5. The value of uncertainty must be a non-negative integer
6. The uncertainty units must be either meters or feet.

We can express all these data constraints using XML Schemas

Validating your data

```
<location>
        <latitude>32.904237</latitude>
        <longitude>73.620290</longitude>
        <uncertainty units="meters">2</uncertainty>
</location>
```

XML

Declare a location element. Require that its content be latitude, longitude, and uncertainty.
Declare a latitude element. Require that its value be between -90 and +90.
Declare a longitude element. Require that its value be between -180 and +180.
Declare a uncertainty element with a units attribute.
Require that the element's value be between 0 and 10.
Require that the attribute's value be either feet or meters.

XML Schema

XML Schema validator

Data is ok!

What does an XML Schema accomplish?

Declare a location element. Require that its content be latitude, longitude, and uncertainty.
Declare a latitude element. Require that its value be between -90 and +90.
Declare a longitude element. Require that its value be between -180 and +180.
Declare a uncertainty element with a units attribute. Require that the element's value be between 0 and 10. Require that the attribute's value be either feet or meters.

XML Schema

Answer:
It creates an XML vocabulary:

<location>, <latitude>, <longitude>, <uncertainty>

It specifies the contents of each element, and the restrictions on the content.

It does one more thing ...

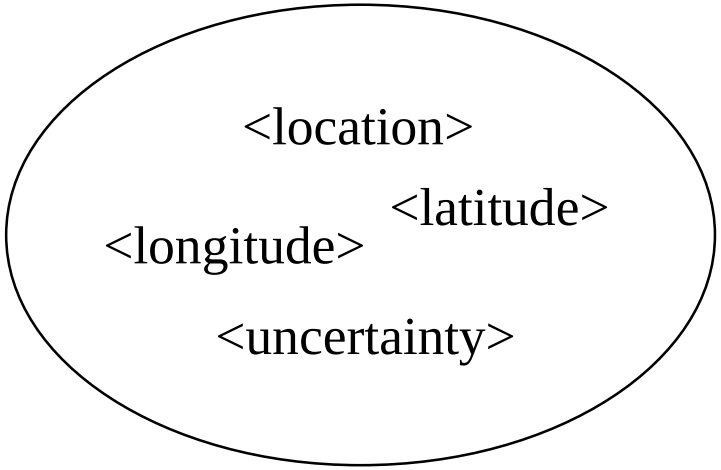
Namespace = http://www.example.org/target
Declare a location element. Require that its content be latitude, longitude, and uncertainty.
Declare a latitude element. Require that its value be between -90 and +90.
Declare a longitude element. Require that its value be between -180 and +180.
Declare a uncertainty element with a units attribute. Require that the element's value be between 0 and 10. Require that the attribute's value be either feet or meters.

XML Schema

An XML Schema specifies that the XML vocabulary that is being created shall be in a "namespace"

http://www.example.org/target Namespace

http://www.example.org/target



Purpose of XML Schemas (and DTDs)

- Specify:
  - the *structure* of actual/instance documents
    - "this element contains these elements, which contains these other elements, etc"
  - the *datatype* of each element/attribute
    - "this element shall hold an integer with the range 0 to 12,000" (DTDs don't do too well with specifying datatypes like this)

Motivation for XML Schemas

- People are dissatisfied with DTDs
  - It's a different syntax
    - You write your XML (instance) document using one syntax and the DTD uses another syntax --> bad, inconsistent
  - Limited datatype capability
    - DTDs support a very limited capability for specifying datatypes. You can't, for example, express "I want the <elevation> element to hold an integer with a range of 0 to 12,000“
    - DTD supports 10 datatypes; XML Schemas supports 44+ datatypes

Highlights of XML Schemas

- XML Schemas are a tremendous advancement over DTDs:
  - Enhanced datatypes
    - 44+ versus 10
    - Can create your own datatypes
      - Example: "This is a new type based on the string type and elements of this type must follow this pattern: ddd-dddd, where 'd' represents a digit".
  - Written in the same syntax as instance documents
    - less syntax to remember
  - Object-oriented'ish
    - Can extend or restrict a type (derive new type definitions on the basis of old ones)
  - Can define the child elements to occur in any order

Example

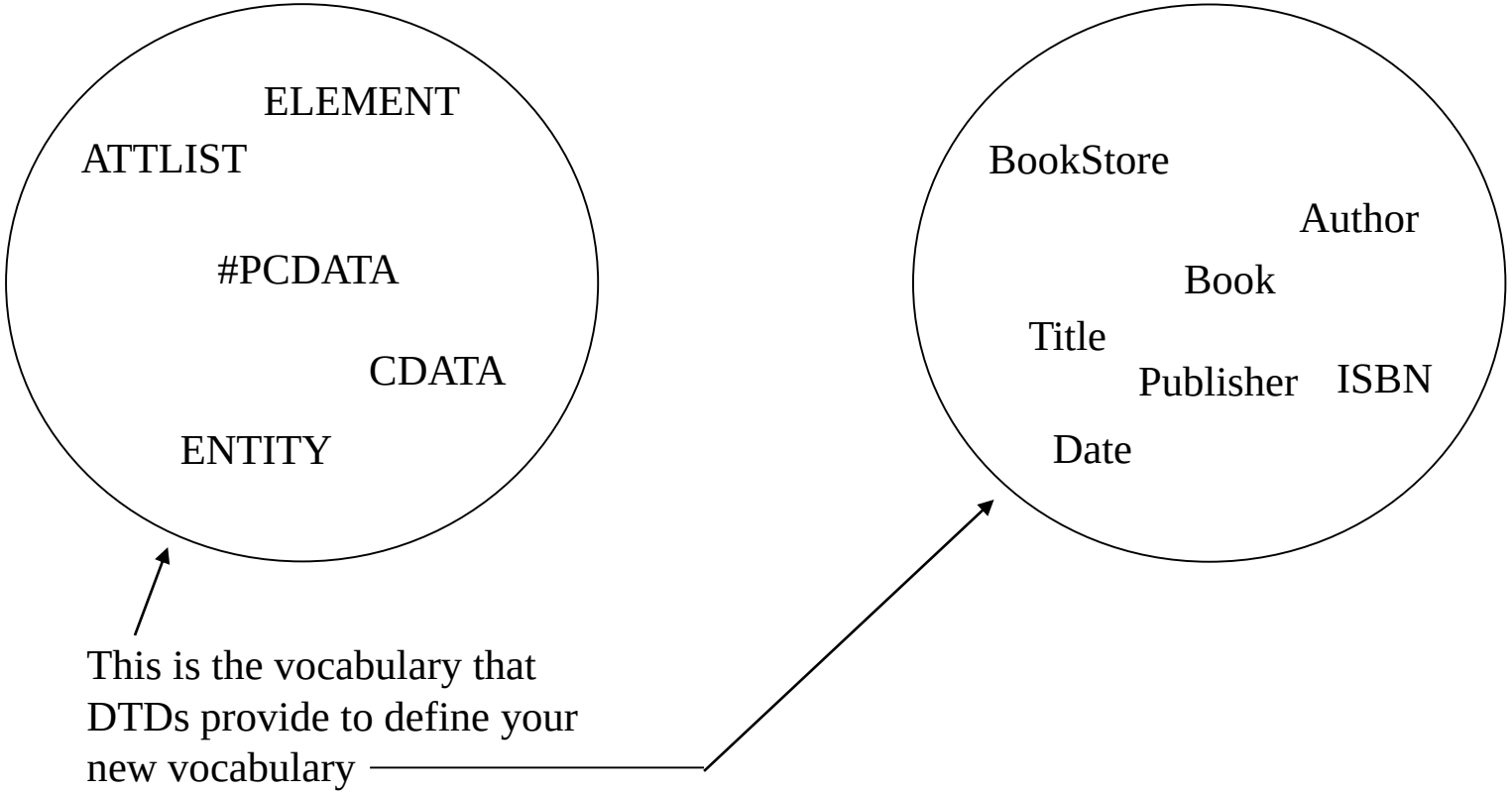
- Convert the BookStore.dtd (next page) to the XML Schema syntax
  - for this first example we will make a straight, one-to-one conversion, i.e., Title, Author, Date, ISBN, and Publisher will hold strings, just like is done in the DTD
  - We will gradually modify the XML Schema to use stronger types

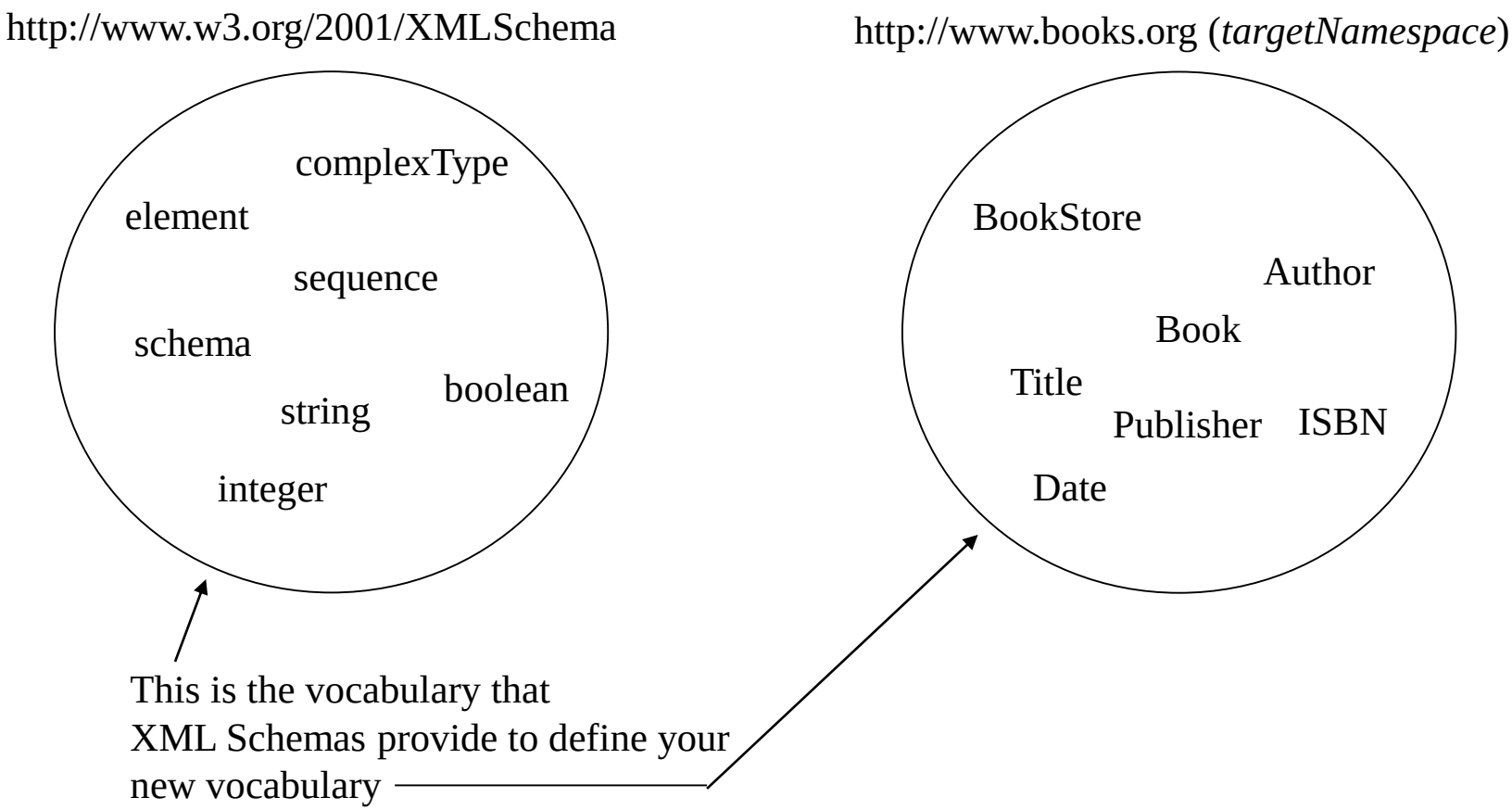
BookStore.dtd

```
<!ELEMENT BookStore (Book+)>
<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
```

BookStore XML example

```xml
<?xml version="1.0"?>
<!DOCTYPE BookStore SYSTEM "BookStore.dtd">
<BookStore>
  <Book>
     <Title>My Life and Times</Title>
     <Author>Paul McCartney</Author>
     <Date>July, 1998</Date>
     <ISBN>94303-12021-43892</ISBN>
     <Publisher>McMillin Publishing</Publisher>
  </Book>
</BookStore>
```


ELEMENT
ATTLIST
#PCDATA
CDATA
ENTITY
BookStore
Author
Book
Title
Publisher
ISBN
Date
This is the vocabulary that
DTDs provide to define your
new vocabulary



One difference between XML Schemas and DTDs is that the XML Schema vocabulary is associated with a name (namespace). Likewise, the new vocabulary that you define must be associated with a name (namespace). With DTDs neither set of vocabularies is associated with a name (namespace) [because DTDs pre-dated namespaces].

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

(explanations on succeeding pages)

BookStore.xsd

xsd = Xml-Schema Definition

BookStore XML example

```
<?xml version="1.0"?>
<BookStore xmlns ="http://www.books.org"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://www.books.org BookStore.xsd">
  <Book>
     <Title>My Life and Times</Title>
     <Author>Paul McCartney</Author>
     <Date>July, 1998</Date>
     <ISBN>94303-12021-43892</ISBN>
     <Publisher>McMillin Publishing</Publisher>
  </Book>
</BookStore>
```

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Equivalent in DTD

↓

```
<!ELEMENT BookStore (Book+)>

<!ELEMENT Book (Title, Author, Date,
                ISBN, Publisher)>

<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
```

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

All XML Schemas have "schema" as the root element.

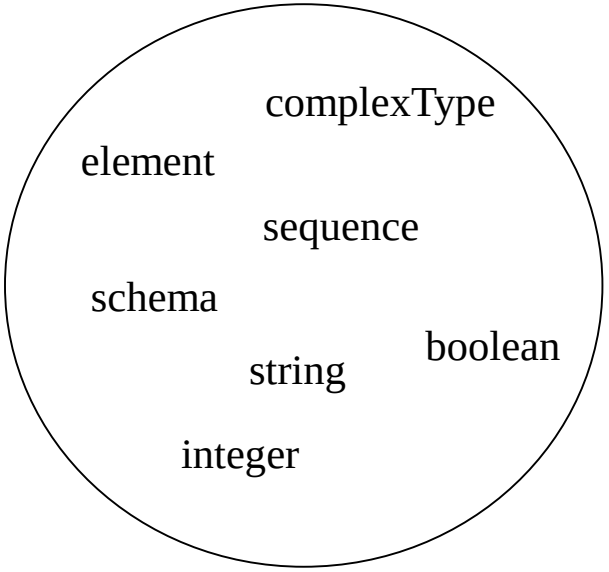
```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

The elements and datatypes that are used to construct schemas
- schema
- element
- complexType
- sequence
- string

come from the http://…/XMLSchema namespace

XMLSchema Namespace

http://www.w3.org/2001/XMLSchema



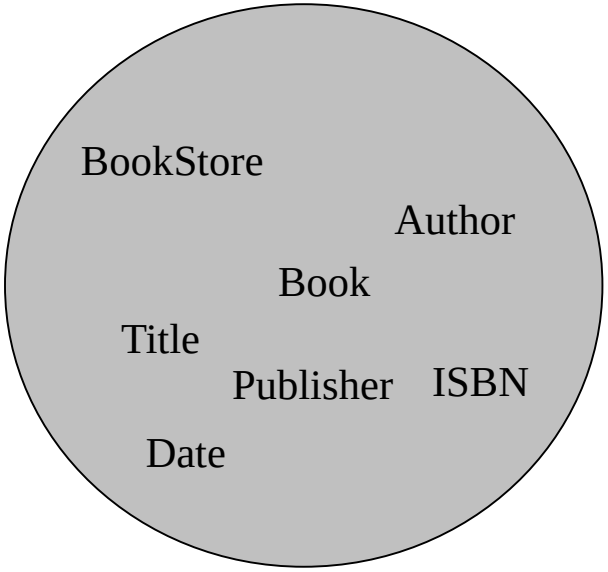
```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Indicates that the elements defined by this schema
- BookStore
- Book
- Title
- Author
- Date
- ISBN
- Publisher

are to go in the http://www.books.org namespace

Book Namespace (targetNamespace)

http://www.books.org (targetNamespace)



```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

The default namespace is http://www.books.org which is the targetNamespace!

This is referencing a Book element declaration. The Book in what namespace? Since there is no namespace qualifier it is referencing the Book element in the default namespace, which is the targetNamespace! Thus, this is a reference to the Book element declaration in this schema.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

This is a directive to any instance documents which conform to this schema: Any elements used by the instance document which were declared in this schema must be namespace qualified.

Referencing a schema in an XML instance document

```
<?xml version="1.0"?>
<BookStore xmlns ="http://www.books.org" (1)
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" (3)
           xsi:schemaLocation="http://www.books.org
                               BookStore.xsd"> (2)
    <Book>
        <Title>My Life and Times</Title>
        <Author>Paul McCartney</Author>
        <Date>July, 1998</Date>
        <ISBN>94303-12021-43892</ISBN>
        <Publisher>McMillin Publishing</Publisher>
    </Book>
    ...
</BookStore>
```

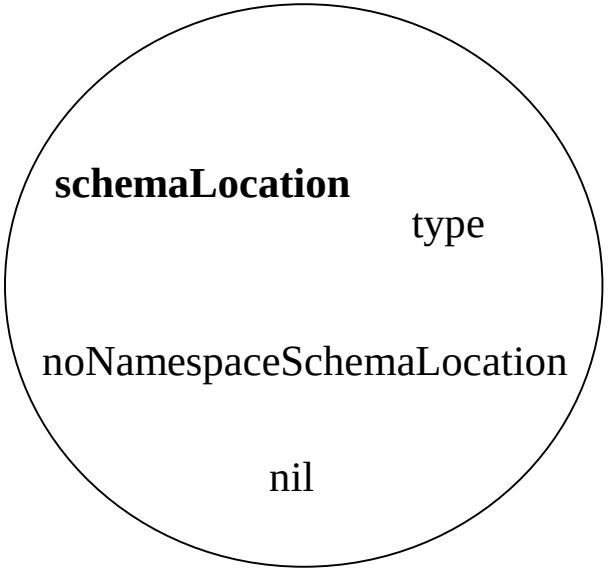
1. First, using a default namespace declaration, tell the schema-validator that all of the elements used in this instance document come from the *http://www.books.org* namespace.

2. Second, with *schemaLocation* tell the schema-validator that the *http://www.books.org* namespace is defined by *BookStore.xsd* (note: *schemaLocation* contains a pair of values).

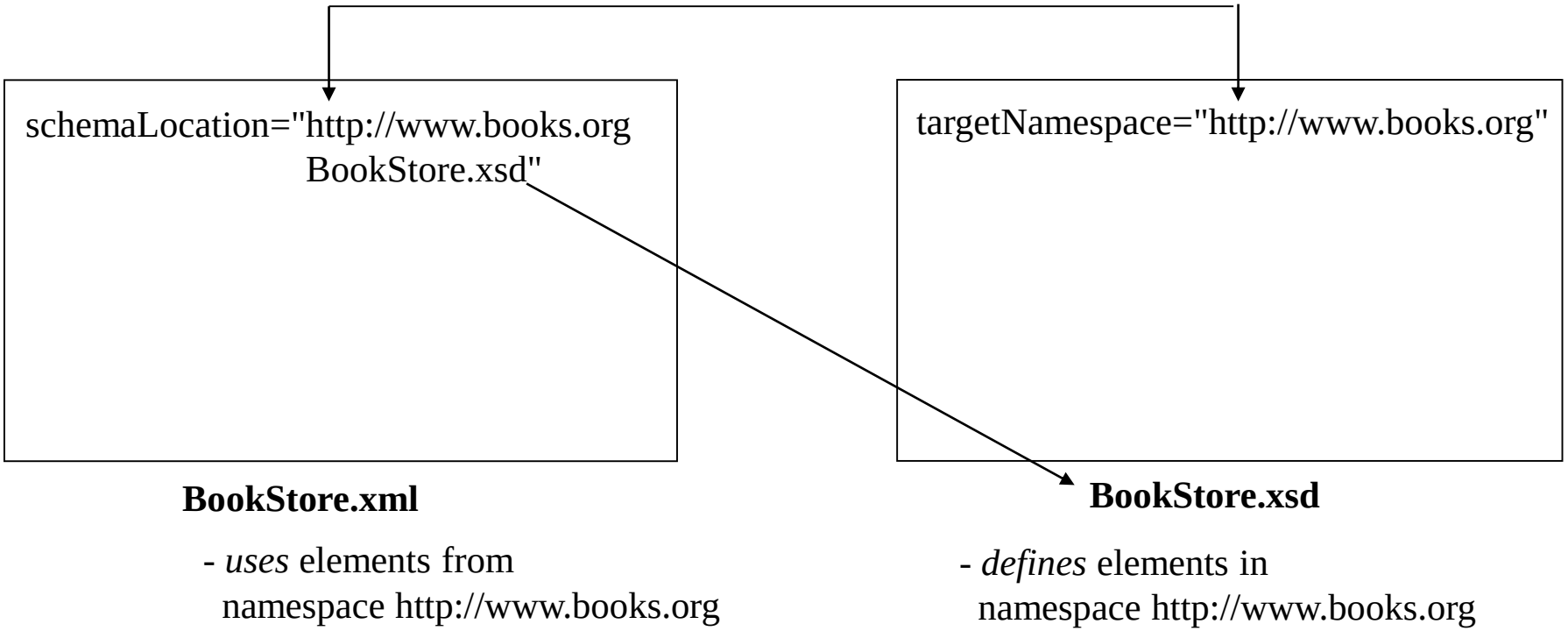
3. Third, tell the schema-validator that the *schemaLocation* attribute we are using is the one in the *XMLSchema-instance* namespace.

XMLSchema-instance Namespace

http://www.w3.org/2001/XMLSchema-instance

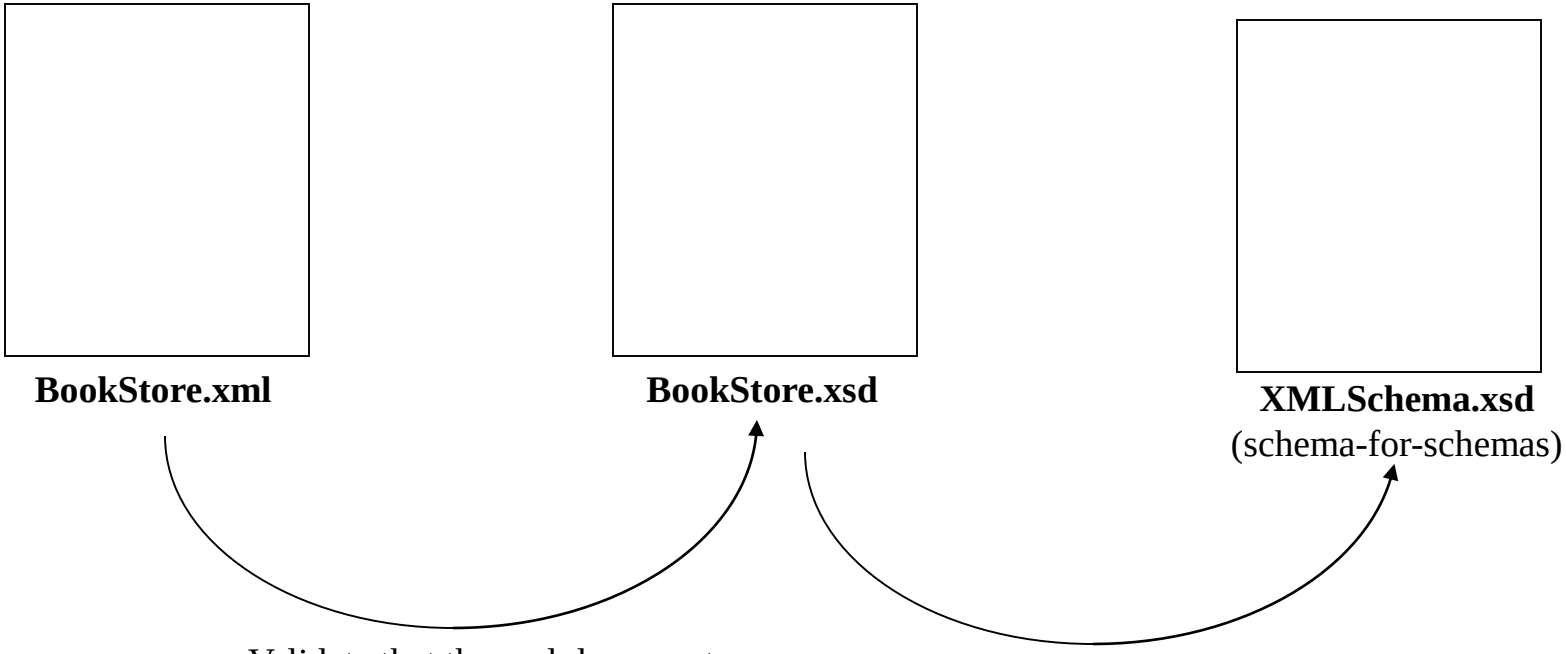


Referencing a schema in an XML instance document



A schema *defines* a new vocabulary. Instance documents *use* that new vocabulary.

Note multiple levels of checking



Validate that the xml document conforms to the rules described in BookStore.xsd

Validate that BookStore.xsd is a valid schema document, i.e., it conforms to the rules described in the schema-for-schemas

Default Value for minOccurs and maxOccurs

- The default value for minOccurs is "1"
- The default value for maxOccurs is "1"

Equivalent!

```
<xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
```

```
<xsd:element ref="Title"/>
```

No targetNamespace (noNamespaceSchemaLocation)

- Sometimes you may wish to create a schema but without associating the elements with a namespace.
- The *targetNamespace* attribute is actually an optional attribute of <schema>. Thus, if you don't want to specify a namespace for your schema then simply don't use the *targetNamespace* attribute.
- Consequences of having no namespace
  - 1. In the instance document don't namespace qualify the elements.
  - 2. In the instance document, instead of using *schemaLocation* use *noNamespaceSchemaLocation*.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">
   <xsd:element name="BookStore">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="Book" minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="Book">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="Title"/>
            <xsd:element ref="Author"/>
            <xsd:element ref="Date"/>
            <xsd:element ref="ISBN"/>
            <xsd:element ref="Publisher"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="Title" type="xsd:string"/>
   <xsd:element name="Author" type="xsd:string"/>
   <xsd:element name="Date" type="xsd:string"/>
   <xsd:element name="ISBN" type="xsd:string"/>
   <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Note that there is no *targetNamespace* attribute, and note that there is no longer a default namespace.

```
<?xml version="1.0"?>
<BookStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation= "BookStore.xsd">
      <Book>
            <Title>My Life and Times</Title>
            <Author>Paul McCartney</Author>
            <Date>1998</Date>
            <ISBN>1-56592-235-2</ISBN>
            <Publisher>McMillin Publishing</Publisher>
      </Book>
      …
</BookStore>
```

1. Note that there is no default namespace declaration. So, none of the elements are associated with a namespace.
2. Note that we do not use *xsi:schemaLocation* (since it requires a pair of values - a namespace and a URL to the schema for that namespace). Instead, we use *xsi:noNamespaceSchemaLocation*.

Assembling an Instance Document from Multiple Schema Documents

- An instance document may be composed of elements from multiple schemas.
- Validation can apply to the entire XML instance document, or to a single element.

```
<?xml version="1.0"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation=
               "http://www.book.org
                Book.xsd
                http://www.employee.org
                Employee.xsd">
  <Books>
    <Book xmlns="http://www.book.org">
          <Title>My Life and Times</Title>
          <Author>Paul McCartney</Author>
          <Date>1998</Date>
          <ISBN>1-56592-235-2</ISBN>
          <Publisher>Macmillan Publishing</Publisher>
    </Book>
    <Book xmlns="http://www.book.org">
          <Title>Illusions: The Adventures of a Reluctant Messiah</Title>
          <Author>Richard Bach</Author>
          <Date>1977</Date>
          <ISBN>0-440-34319-4</ISBN>
          <Publisher>Dell Publishing Co.</Publisher>
    </Book>
    <Book xmlns="http://www.book.org">
          <Title>The First and Last Freedom</Title>
          <Author>J. Krishnamurti</Author>
          <Date>1954</Date>
          <ISBN>0-06-064831-7</ISBN>
          <Publisher>Harper &amp; Row</Publisher>
    </Book>
  </Books>
  <Employees>
    <Employee xmlns="http://www.employee.org">
      <Name>John Doe</Name>
      <SSN>123-45-6789</SSN>
    </Employee>
    <Employee xmlns="http://www.employee.org">
      <Name>Sally Smith</Name>
      <SSN>000-11-2345</SSN>
    </Employee>
  </Employees>
</Library>
```

Library.xml

Validating against two schemas

The <Book> elements are defined in Book.xsd, and the <Employee> elements are defined in Employee.xsd. The <Library>, <Books>, and <Employees> elements are not defined in any schema!

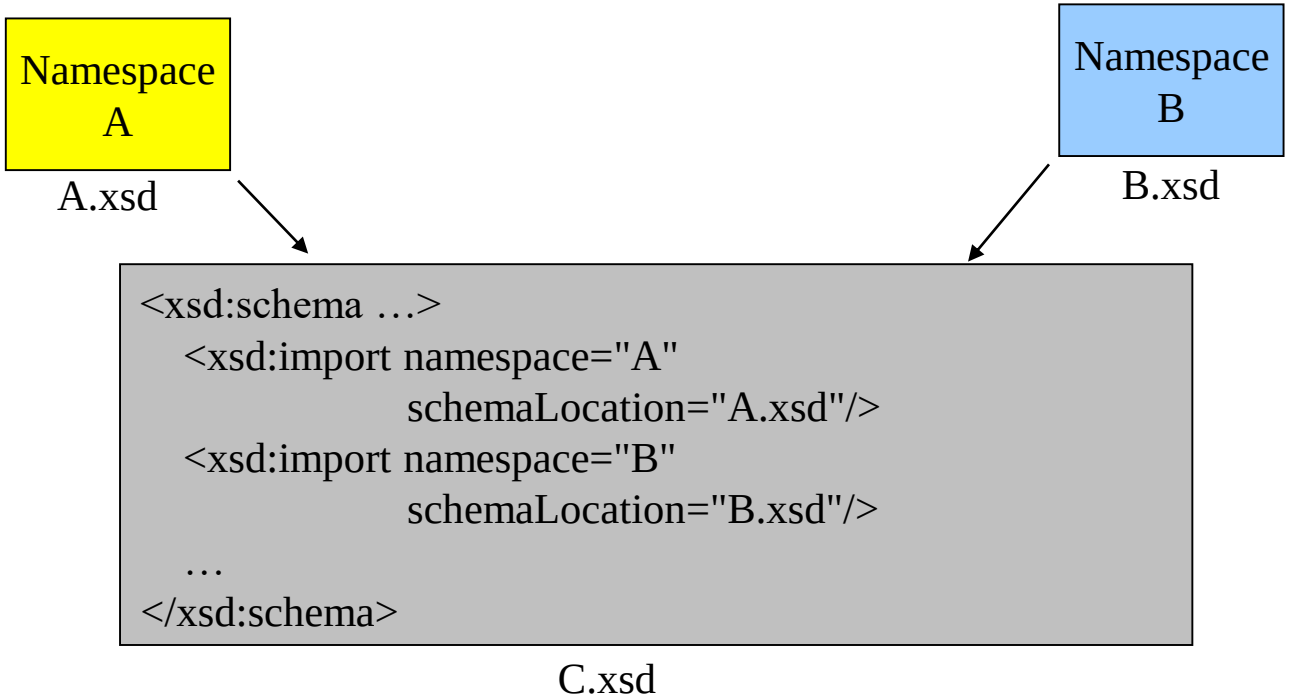
1. A schema validator will validate each Book element against Book.xsd.
2. It will validate each Employee element against Employee.xsd.
3. It will not validate the other elements (we'll see how to do this later)

Lax Validation vs Strict Validation

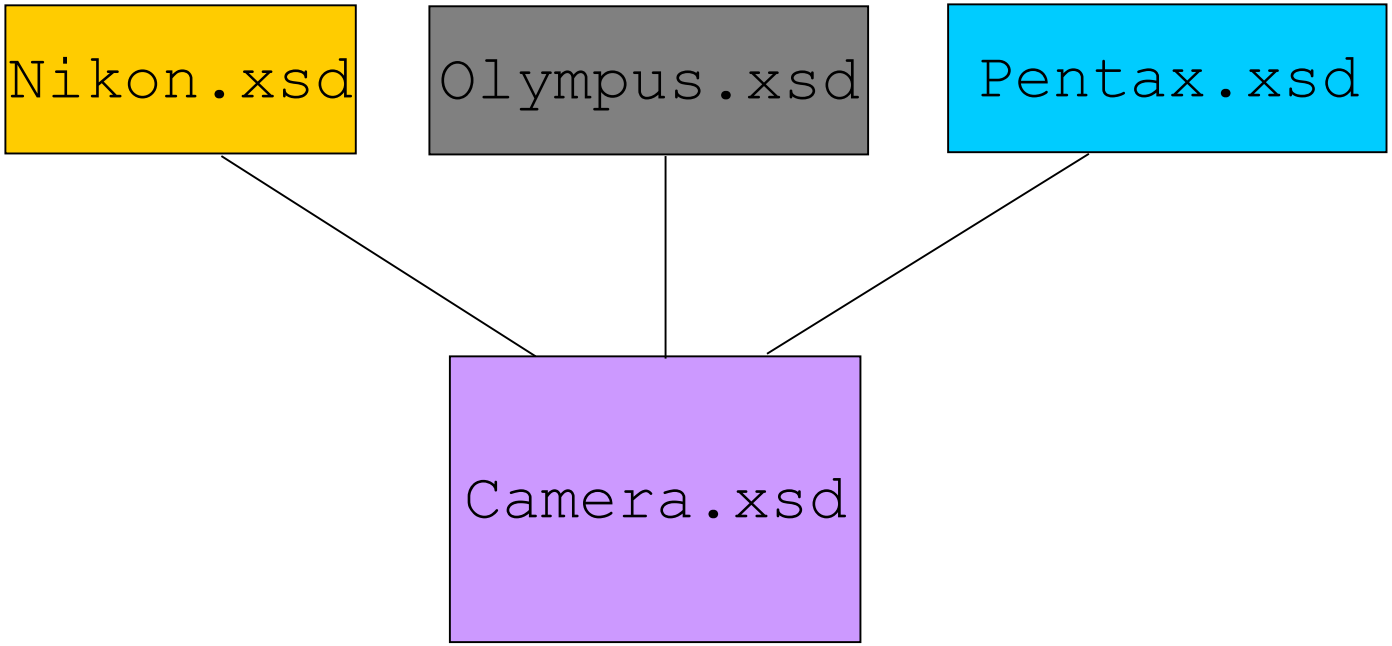
- On the previous slide there were elements Library, Books, and Employees. Employees had no schema to validate against.
- Lax validation is where the schema validator skips over elements for which no schema is available.
- Strict validation is where the schema validator requires validation of every element
- The W3C validator “XSV” performs lax validation. Thus, it will accept the instance document on the previous slide (but it will note validation="lax" in its output)
- Other validators may differ. Consequently, they could reject the instance document on the previous slide.

Assembling a Schema from Multiple Schema Documents with Different Namespaces

- The **import** element allows you to access elements and types in a **different namespace**



Camera Schema



Nikon.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.nikon.com"
            xmlns="http://www.nikon.com"
            elementFormDefault="qualified">
  <xsd:complexType name="body_type">
    <xsd:sequence>
      <xsd:element name="description" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Olympus.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.olympus.com"
      xmlns="http://www.olympus.com"
      elementFormDefault="qualified">
  <xsd:complexType name="lens_type">
    <xsd:sequence>
      <xsd:element name="zoom" type="xsd:string"/>
      <xsd:element name="f-stop" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Pentax.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.pentax.com"
      xmlns="http://www.pentax.com"
      elementFormDefault="qualified">
  <xsd:complexType name="manual_adapter_type">
    <xsd:sequence>
      <xsd:element name="speed" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.camera.org"
            xmlns:nikon="http://www.nikon.com"
            xmlns:olympus="http://www.olympus.com"
            xmlns:pentax="http://www.pentax.com"
            elementFormDefault="qualified">
  <xsd:import namespace="http://www.nikon.com"
              schemaLocation="Nikon.xsd"/>
  <xsd:import namespace="http://www.olympus.com"
              schemaLocation="Olympus.xsd"/>
  <xsd:import namespace="http://www.pentax.com"
              schemaLocation="Pentax.xsd"/>
  <xsd:element name="camera">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="body" type="nikon:body_type"/>
        <xsd:element name="lens" type="olympus:lens_type"/>
        <xsd:element name="manual_adapter" type="pentax:manual_adapter_type"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
<xsd:schema>
```

Defining prefixes for namespaces

These import elements give us access to the components in these other schemas.

Here we use the *body_type* that is defined in the Nikon namespace

Camera.xsd

```
<?xml version="1.0"?>
<c:camera xmlns:c="http://www.camera.org"
          xmlns:nikon="http://www.nikon.com"
          xmlns:olympus="http://www.olympus.com"
          xmlns:pentax="http://www.pentax.com"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation=
              "http://www.camera.org
               Camera.xsd
               http://www.nikon.com
               Nikon.xsd
               http://www.olympus.com
               Olympus.xsd
               http://www.pentax.com
               Pentax.xsd">
  <c:body>
    <nikon:description>Ergonomically designed casing for easy handling</nikon:description>
  </c:body>
  <c:lens>
    <olympus:zoom>300mm</olympus:zoom>
    <olympus:f-stop>1.2</olympus:f-stop>
  </c:lens>
  <c:manual_adapter>
    <pentax:speed>1/10,000 sec to 100 sec</pentax:speed>
  </c:manual_adapter>
</c:camera>
```

The Camera instance uses elements from the Nikon, Olympus, and Pentax namespaces.

Camera.xml

Redundant!

- On the previous slide, the value of schemaLocation contained four pairs of values - one for camera, and three for each schema that it uses. The later three are redundant. Once you give the schema-validator the URL to the camera schema it will examine the camera schema and see the import elements, thus it will deduce the other schemas being used (Nikon, Olympus, and Pentax)

- The next slide shows the non-redundant version.

```
<?xml version="1.0"?>
<c:camera xmlns:c="http://www.camera.org"
          xmlns:nikon="http://www.nikon.com"
          xmlns:olympus="http://www.olympus.com"
          xmlns:pentax="http://www.pentax.com"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation=
                "http://www.camera.org
                 Camera.xsd">
   <c:body>
      <nikon:description>Ergonomically designed casing for easy handling</nikon:description>
   </c:body>
   <c:lens>
      <olympus:zoom>300mm</olympus:zoom>
      <olympus:f-stop>1.2</olympus:f-stop>
   </c:lens>
   <c:manual_adapter>
      <pentax:speed>1/10,000 sec to 100 sec</pentax:speed>
   </c:manual_adapter>
</c:camera>
```

Camera.xml (non-redundant version)