Enhanced Control using Fine-Grain Validation

Roger L. Costello
July 19, 2013

Course-Grain Validation Limits Your Options

When an XML instance document is validated against an XML Schema, the validator returns valid or invalid. If invalid is returned, the XML document is rejected. If valid is returned, the XML document is accepted and processed. The decision to process inputs is made on a per-document basis. That *course-grain*, all-or-nothing approach to accepting and processing inputs is very limiting. Suppose, for example, that the XML document contains data for 1000 Books and all the Books are valid except one. Rejecting the entire document because of one bad Book is terribly inefficient.

This paper shows how to validate – in an automated fashion – each item of an XML document and make validation decisions on a per-item basis rather than a per-document basis. The items to be validated may be as fine-grain or as course-grain as desired: the item may be a single element (or even just the text within a leaf element) or the item may be an element composed of many descendent elements. Validation may be done from the outside in (the root element to the leaf nodes) or the inside out (the leaf nodes to the root element).

I call the technique presented in this paper: *fine-grain validation*.

By using the technique, your XML Schemas change from being a large template that XML documents must conform to, to a collection of rules from which we can pick and choose.

Fine-Grain Validation Expands Your Options

Continuing with the example of an XML document containing data for 1000 Books, suppose that each Book contains (among other things) data for its ISBN and suppose that an invalid ISBN is encountered. A decision must be made as to what to do. Here are some possible options:

1. Fix the ISBN so that it is valid.
2. Replace the erroneous value with a placeholder.
3. Delete the data and leave the ISBN element empty, i.e., <ISBN />
4. Delete the ISBN element.
5. Delete the Book that the ISBN element resides within.

That is fantastic! We have lots of options for handling invalid data.

Here is the Technique

Step 1: create an XML Schema. Use the Salami Slice design (make everything global); this will give maximum flexibility in terms of validating large items or small items. At the bottom of this paper is the XML Schema for the example we have been considering, BookStore.xsd

Step 2: create an XML instance document. See BookStore.xml below.

Step 3: create a *schema-aware XSLT program*. At the top of the XSLT program import the XML Schema:

```
<xsl:import-schema schema-location="BookStore.xsd"/>
```

Step 4: create a (template) rule for each element and attribute. Here is a rule for the ISBN element. The rule tests the value of ISBN to see if it is of type ISBN-type (ISBN-type is a simpleType in the XML Schema):

```
<xsl:template match="ISBN">
    <xsl:if test=". castable as ISBN-type">
        <ISBN><xsl:value-of select="."/></ISBN>
    </xsl:if>
</xsl:template>
```

That rule says: If the value of ISBN is schema-valid then output the ISBN, otherwise don't output anything (i.e., remove the ISBN element).

Wow! That is powerful. No longer are we limited to making accept/reject decisions based on the entire document. This rule shows that we can make accept/reject decisions based on an individual element.

Here is another rule. If the Date element does not contain a valid year, then remove the Date element:

```
<xsl:template match="Date">
    <xsl:if test=". castable as xsd:gYear">
        <Date><xsl:value-of select="."/></Date>
    </xsl:if>
</xsl:template>
```

The above two rules chose to remove the elements if they are not schema-valid. Remember, however, that you have lots of options on what to do if an item is not schema-valid.

Below is the schema-aware XSLT program, BookStore.xsl

Note: If you are using oXygen XML to perform schema-aware processing of XML instance documents, you will find this tip helpful. In the menu: `Options / Preferences`

In the dialog box for `Saxon-HE/PE/EE` is a section titled `Saxon-EE specific options`. In there is this: `Validation of the source file ("-val"):`

Set that to: `Disable schema validation`. Now your XSLT program will behave as desired.

BookStore.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <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" minOccurs="0" />
                <xsd:element ref="Author" minOccurs="0" />
                <xsd:element ref="Date" minOccurs="0" />
                <xsd:element ref="ISBN" minOccurs="0" />
                <xsd:element ref="Publisher" minOccurs="0" />
            </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:gYear"/>
    <xsd:element name="ISBN" type="ISBN-type"/>
    <xsd:element name="Publisher" type="xsd:string"/>

    <xsd:simpleType name="ISBN-type">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="\d{1}-\d{5}-\d{3}-\d{1}|\d{1}-\d{3}-\d{5}-
\d{1}|\d{1}-\d{2}-\d{6}-\d{1}"/>
        </xsd:restriction>
    </xsd:simpleType>

</xsd:schema>
```

BookStore.xml

This XML document contains some invalid data. The schema-aware XSLT program will remove the invalid items and output an XML document with valid-only items.

```
<?xml version="1.0"?>
<BookStore>
        <Book>
                <Title>My Life and Times</Title>
                <Author>Paul McCartney</Author>
                <Date>1998</Date>
                <ISBN>xxx1-56592-235-2</ISBN>             <!-- Invalid -->
                <Publisher>McMillan Publishing</Publisher>
        </Book>
        <Book>
                <Title>Illusions</Title>
                <Author>Richard Bach</Author>
                <Date>1977</Date>
                <ISBN>0-440-34319-4</ISBN>
                <Publisher>Dell Publishing Co.</Publisher>
        </Book>
        <Book>
                <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>
                <Review>Wonderful book!</Review>          <!-- Invalid -->
        </Book>
        <Magazine>                                        <!-- Invalid -->
                <Title>Scientific American</Title>
        </Magazine>
</BookStore>
```

BookStore.xsl

```
<?xml version="1.0"?>
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               exclude-result-prefixes="xsd"
               version="2.0">

    <xsl:output method="xml"/>

    <xsl:import-schema schema-location="BookStore.xsd"/>

    <!-- If the root element is not BookStore, that
         is a fatal error -->
    <xsl:template match="/*[not(self::BookStore)]" >
        <xsl:value-of select="error((),'Hey! The root element is not
BookStore!')" />
    </xsl:template>

    <!-- Copy BookStore verbatim -->
    <xsl:template match="BookStore">
        <xsl:copy>
            <xsl:copy-of select="@*" />
            <xsl:apply-templates />
        </xsl:copy>
    </xsl:template>

    <!-- Copy BookStore verbatim -->
    <xsl:template match="Book">
        <xsl:copy>
            <xsl:copy-of select="@*" />
            <xsl:apply-templates />
        </xsl:copy>
    </xsl:template>

    <!-- Delete any children of BookStore that are not Books -->
    <xsl:template match="BookStore/*[not(self::Book)]" />

    <!-- If the value of ISBN is invalid, delete the
         ISBN element -->
    <xsl:template match="ISBN">
        <xsl:if test=". castable as ISBN-type">
            <ISBN><xsl:value-of select="."/></ISBN>
        </xsl:if>
    </xsl:template>

    <!-- If the value of Date is not a year, delete the
         Date element -->
    <xsl:template match="Date">
        <xsl:if test=". castable as xsd:gYear">
```

```
            <Date><xsl:value-of select="."/></Date>
        </xsl:if>
    </xsl:template>

    <!-- Copy Title, Author, and Publisher verbatim -->
    <xsl:template match="Title | Author | Publisher">
        <xsl:copy>
            <xsl:copy-of select="@*" />
            <xsl:apply-templates />
        </xsl:copy>
    </xsl:template>

    <!-- Delete any children of Book that are not expected -->
    <xsl:template match="Book/*[not(self::Title or self::Author or self::Date
or self::ISBN or self::Publisher)]" />

</xsl:transform>
```

Acknowledgements

Thanks to the following people for helping me understand and implement this technique:

- David Carlisle
- Ken Holman
- Martin Honnen
- Michael Kay
- Andrew Welch