PB138 — Advanced XML Processing

Outline

- Types of transformations
- XML pipelining

XML Transformations

- An XML transformation language is a programming language designed specifically to transform an input XML document to an output document which satisfies some specific goal.
- There are two special cases of transformation:
  - *XML to XML* — the output document is an XML document.
  - *XML to Data* — the output document is a byte stream.

XML Pipeline

- *XML Pipeline* connects of XML processes, especially *XML transformations* and *XML validations*.
- For instance, given two transformations `T1` and `T2`, the two can be connected so that:
  a. input XML document
  b. is transformed by `T1` and then
  c. output of `T1` is fed as input document to `T2`
- Simple pipelines like the one described above are called *linear*: a single input document always goes through the same sequence of transformations to produce a single output document.

XML Pipeline operations

- *Linear* operations
- *Non-linear* operations

Linear operations

- *Micro* operations
- *Document* operations
- *Sequence* operations

Linear: Micro-operations

- Operate at the inner document level:
  - `Rename` — renames elements or attributes without modifying the content

- `Replace` — replaces elements or attributes
- `Insert` — adds a new data element to the output stream at a specified point
- `Delete` — removes an element or attribute (also known as pruning the input tree)
- `Wrap` — wraps elements with additional elements
- `Reorder` — changes the order of elements

Linear: Document operations

They take the input document as a whole:

- `Identity transform` — makes a verbatim copy of its input to the output
- `Compare` — it takes two documents and compare them
- `Transform` — execute a transform on the input file using a specified XSLT file
- `Split` — take a single XML document and split it into distinct documents

Linear: Sequence operations

They are mainly introduced in XProc and help to handle the sequence of documents as a whole:

- `Count` — it takes a sequence of documents and counts them
- `Identity transform` — makes a verbatim copy of its input sequence of documents to the output
- `Split-sequence` — takes a sequence of documents as input and routes them to different outputs depending on matching rules
- `Wrap-sequence` — takes a sequence of documents as input and wraps them into one or more documents

Non-linear operations

- `Conditionals` — where a given transformation is executed if a condition is met while another transformation is executed otherwise
- `Loops` — where a transformation is executed on each node of a node set selected from a document or a transformation is executed until a condition evaluates to `false`
- `Tees` — where a document is fed to multiple transformations potentially happening in parallel
- `Aggregations` — where multiple documents are aggregated into a single document
- `Exception Handling` — where failures in processing can result an alternate pipeline being processed

What is XProc?

- http://www.xfront.com/xproc/

- XProc is an XML Pipeline Language, thus an XML Pipeline implementation
- XProc enables you to declaratively express the activities you want to perform on XML documents
- XProc is a W3C recommendation https://www.w3.org/TR/xproc/

Benefits of XProc

- XProc takes care of orchestrating all the activities
- XProc is a standard way of expressing processing activities
- Since an XProc document is an XML document, you can send it around, transform it, mine it, store it, just like any other XML document

XProc Example - identity

```
<?xml version="1.0" encoding="UTF-8"?>
<p:declare-step xmlns:p="http://www.w3.org/ns/xproc" xmlns:c=
"http://www.w3.org/ns/xproc-step" version="1.0">
   <p:input port="source">
      <p:inline>
         <doc>Hello World!</doc>
      </p:inline>
   </p:input>
   <p:output port="result"/>
   <p:identity/>
</p:declare-step>
```

XProc Example - validation

```
<p:declare-step xmlns:p="http://www.w3.org/ns/xproc"
                name="xinclude-and-validate"
                version="1.0">
  <p:input port="source" primary="true"/>
  <p:input port="schemas" sequence="true"/>
  <p:output port="result">
    <p:pipe step="validated" port="result"/>
  </p:output>
  <p:xinclude name="included">
    <p:input port="source">
      <p:pipe step="xinclude-and-validate" port="source"/>
    </p:input>
  </p:xinclude>
  <p:validate-with-xml-schema name="validated">
    <p:input port="source">
      <p:pipe step="included" port="result"/>
    </p:input>
    <p:input port="schema">
      <p:pipe step="xinclude-and-validate" port="schemas"/>
    </p:input>
  </p:validate-with-xml-schema>
</p:declare-step>
```

XProc Example - A validate and transform pipeline

```
<p:pipeline xmlns:p="http://www.w3.org/ns/xproc" version="1.0">
  <p:choose>
    <p:when test="/*[@version &lt; 2.0]">
      <p:validate-with-xml-schema>
        <p:input port="schema">
          <p:document href="v1schema.xsd"/>
        </p:input>
      </p:validate-with-xml-schema>
    </p:when>
    <p:otherwise>
      <p:validate-with-xml-schema>
        <p:input port="schema">
          <p:document href="v2schema.xsd"/>
        </p:input>
      </p:validate-with-xml-schema>
    </p:otherwise>
  </p:choose>
  <p:xslt>
    <p:input port="stylesheet">
      <p:document href="stylesheet.xsl"/>
    </p:input>
  </p:xslt>
</p:pipeline>
```

XProc Processors

- XML Calabash (not to be confused with Android/iOS-Calabash)
- Calumet
- Tubular
- xmlsh

XML Calabash

- XML Calabash Reference

Resource on XML Pipeline

- XML Pipeline @Wikipedia
- XProc: An XML Pipeline Language (W3C Specification)
- XProc site
- XProc Tutorial