Optimizing XML Delivery with XProc

Vojtěch Toman
EMC Corporation

XML Prague, March 22, 2009

Agenda

XML Application Development with XProc

EMC Documentum XProc Engine

XML Content Delivery Use Cases

Q/A

XML Application Development with XProc

Manual XML programming

- Difficult, error-prone
- “Cross domain” programming
- Makes XML look scary

XProc: Declarative processing model

- Easy to use, robust
- Focus on WHAT, not on the low-level HOW
- Reduces the amount of manual XML programming
- Less bugs, better maintainability and customizability
- Faster/cheaper application development

XProc – Enabling Other Standards

XProc can act as an enabling technology

- XML standards that require a certain level of XML processing capabilities
- Prime example: XForms

The XRX architecture

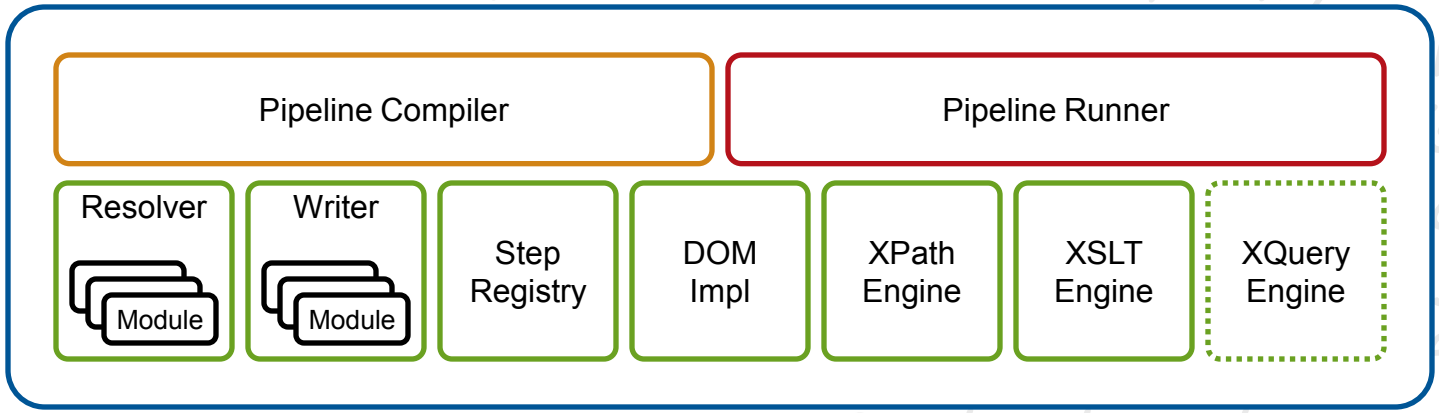
- XForms/REST/X.....
- End-to-end XML model
- XProc is a natural fit

Calumet: EMC Documentum XProc Engine

Implementation of XProc: An XML Pipeline Language

100% Java

Pluggable architecture



Will be released on EMC Developer Network

- Free for developer use
- Part of a wider suite of other XML tools
- Watch http://developer.emc.com

XML Content Delivery

Modern content delivery

- Dynamic publishing
  - Content generation
  - Dynamic content assembly
- Content personalization
  - Driven by user preferences, configuration, or by content itself, …

XML content delivery

- Main data model is XML
- Componentized content, reuse
- Technologies
  - XML, native XML databases, XQuery, XForms, …
  - XProc!

XML Content Delivery Use Cases

Use Case 1: Content Publishing

Use Case 2: Content Assembly

Use Case 3: Content Filtering

Content Publishing: Scenario 1

A pipeline that:

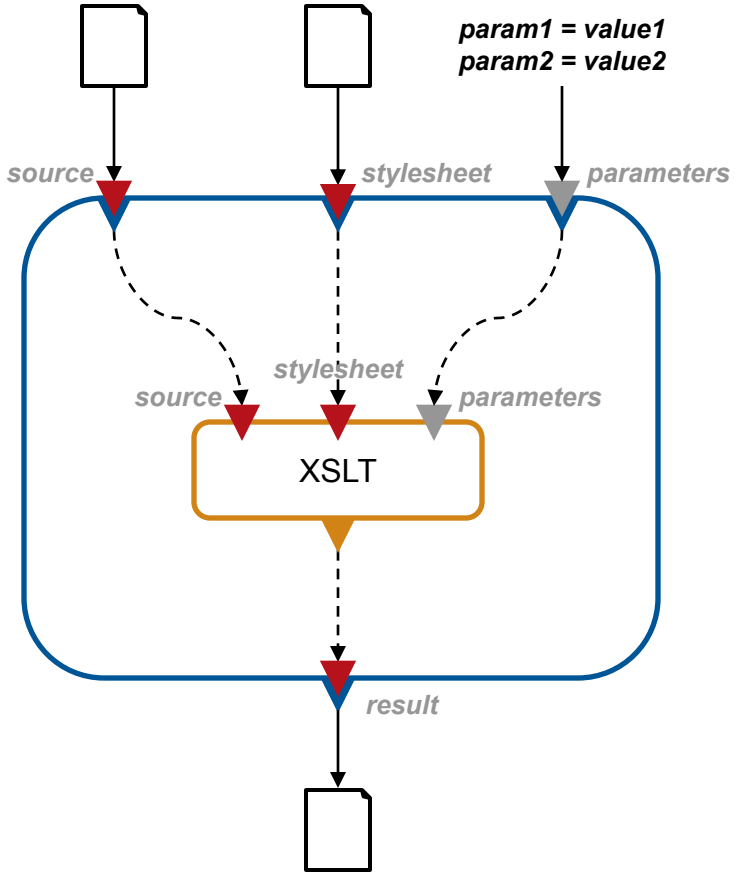
- Applies an XSLT transformation on a document
- *(…the most commonly used XProc pipeline ever?)*

Input:

- XML document
- XSLT stylesheet
- XSLT parameters

Output:

- Transformed document



Content Publishing: Scenario 2

A pipeline that:

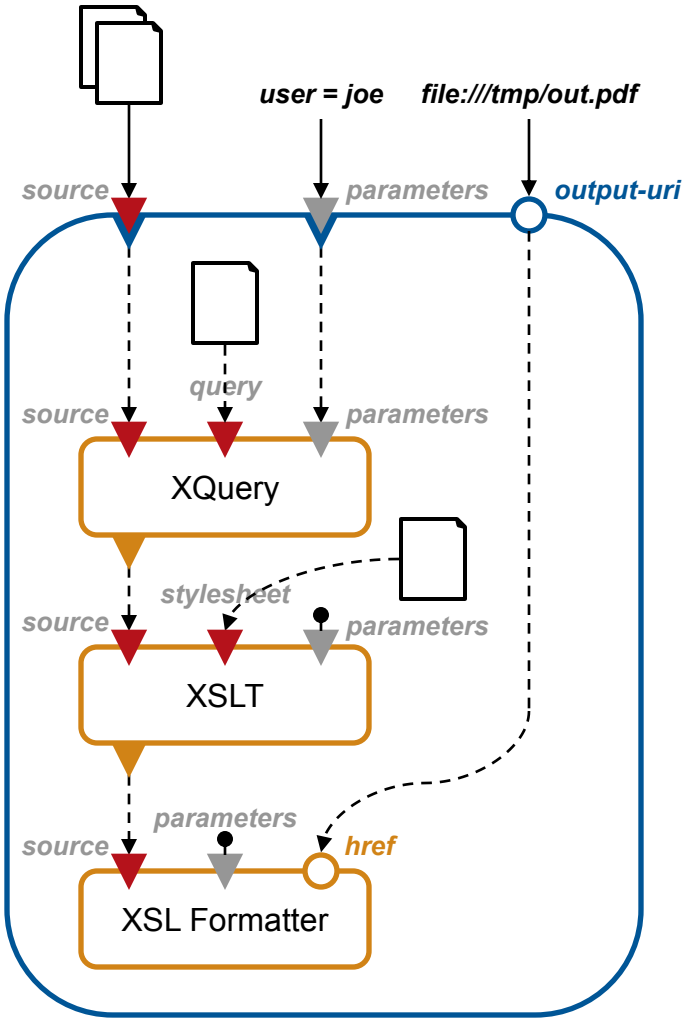
- Executes an XQuery
- Applies an XSLT transformation to produce XSL-FO
- Runs an XSL-FO processor to produce a PDF document

Input:

- Set of XML documents with user comments on element level
- User name
- Target location (URI) of the generated PDF document

Output:

- PDF document with dynamically generated information (comments entered by a particular user)



Content Assembly: Scenario 1

A pipeline that:

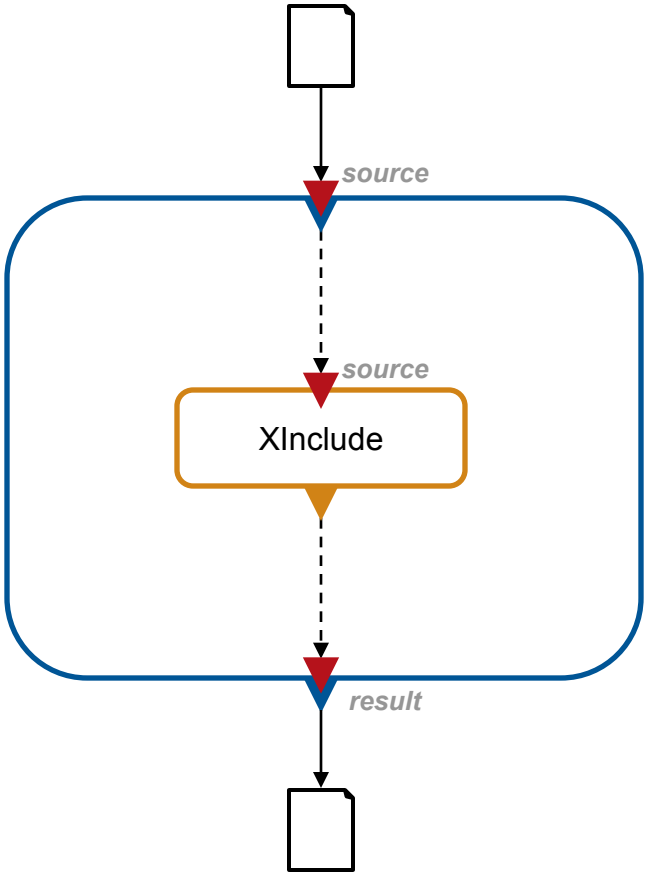
- Performs XInclude processing

Input:

- XML document with XInclude references

Output:

- XML document with all XInclude references resolved



Content Assembly: Scenario 2

A pipeline that:

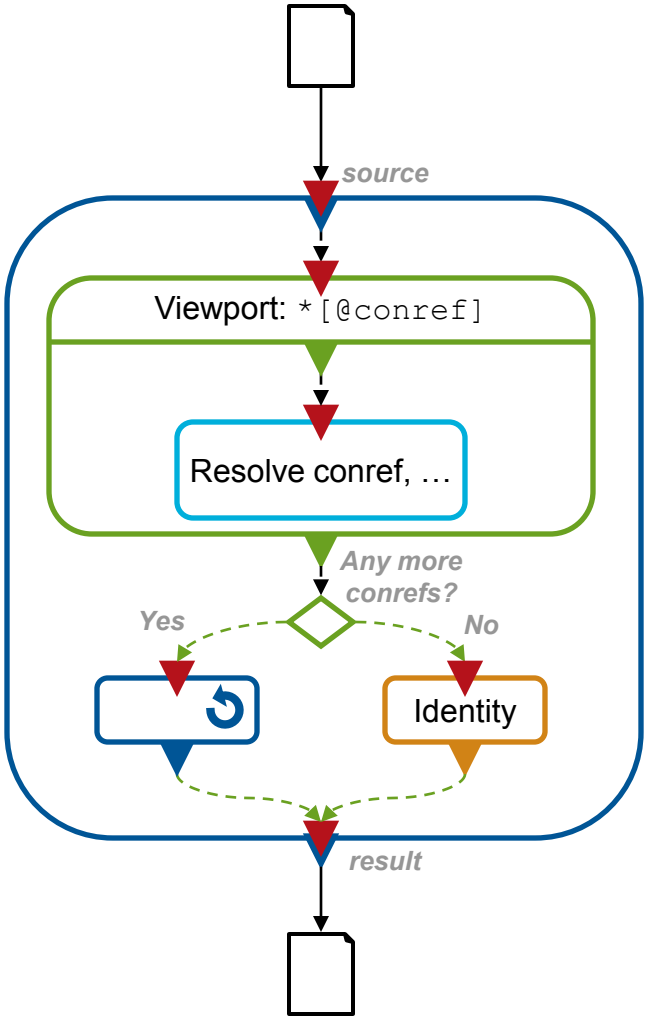
- Resolves (local) DITA conref references

Input:

- DITA topic document with local conrefs

Output:

- DITA topic document with all conrefs resolved



Content Filtering: Scenario 1

A pipeline that:

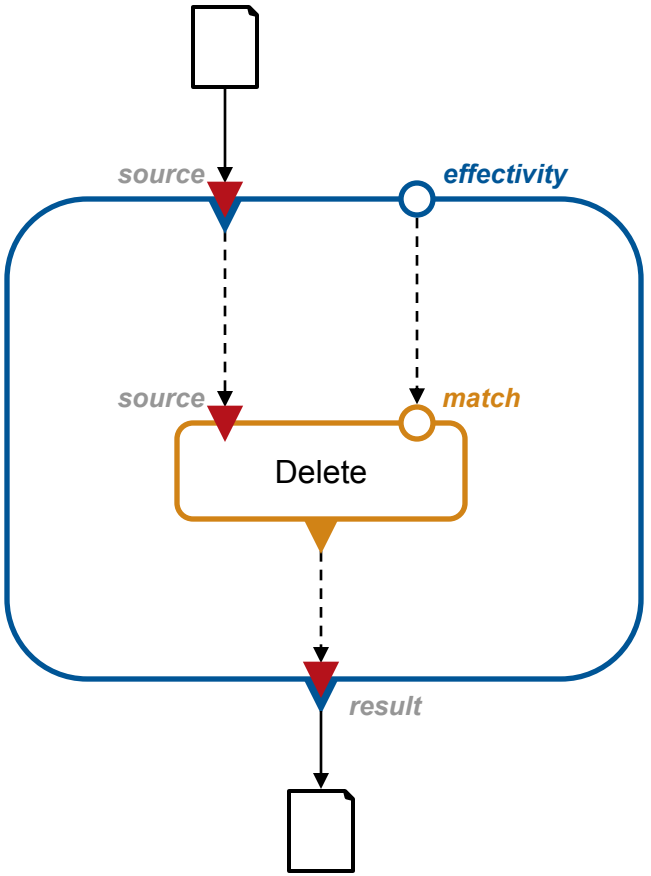
- Performs simple effectivity filtering

Input:

- Docbook document with `condition` attributes (`internal|external`)
- Effectivity information (`internal|external`)

Output:

- Docbook document that contains only information that is effective



Content Filtering: Scenario 2

A pipeline that:

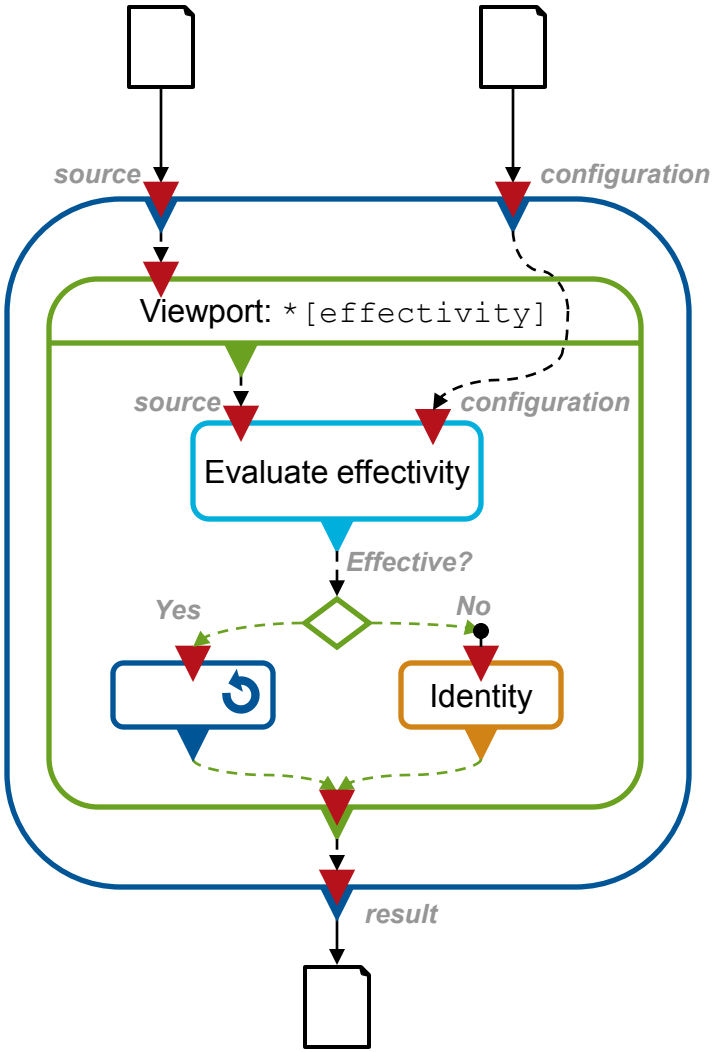
- Performs assertion-based effectivity filtering

Input:

- XML document with effectivity information
- XML document with configuration information

Output:

- XML document that contains only information that is effective



Q/A

Questions and Answers

Vojtěch Toman

toman_vojtech@emc.com

www.emc.com

www.x-hive.com


EMC²
where information lives®