XSL Transform Self-Documentation

Sandy **Tao**

Jeff **Beck**

Demian **Hess**

Abstract

Documenting your own code is an important component in the long-term maintenance of any program. In this project we defined a format and developed a system to automatically self-document the data conversion process at PubMed Central. PubMed Central is a digital archive of full-text biomedical journals. It is developed and managed by NCBI (National Center for Biotechnology Information), a division of the National Library of Medicine.

The XSL stylesheets used at PubMed Central for data conversion present a particular challenge because documentation is needed not only for the reference of developers, but also for digital archivists to ensure that the conversion process conforms to accepted archiving standards. The choices that developers make in writing conversion filters need to be transparent and reviewable. To meet this need, we defined a format for inserting documentation into XSL stylesheets. The documentation had to be easy to maintain and needed to be capable of generating documentation for developers, archivists, and other stakeholders.

As developed, documentation can be inserted into stylesheets either as XML elements or as structured XML comments. The documentation system enables users to log journal- and publisher-specific comments, track revisions, impose a comment hierarchy, and specify the intended audience for each comment.

A set of stylesheets was also written that extracts the documentation and generates HTML for display. Because the system is based on XML and XSLT, it is easily extensible and platform independent—it also has the virtue that it can be used to document itself!

Table of Contents

1. Introduction

PubMed Central [http://www.pubmedcentral.gov/]is a digital archive of full text biomedical journals. It is developed and managed by National Center for Biotechnology Information (NCBI) [http://www.ncbi.nlm.nih.gov/], a division at National Library of Medicine (NLM) [http://www.nlm.nih.gov]. With the new NIH public access policy [http://publicaccess.nih.gov/], PubMed Central is becoming an institutional repository for NIH [http://www.nih.gov/].

As a digital archive, PubMed Central has a responsibility to preserve digital contents over time. The purpose of XSL Transform Self-documentation is to create a self-documentation program for PubMed Central. This tool will help to aid management of the data conversion process, to ensure the quality control and long term access to the digital contents.

This program allows the developers to create useful documentation very easily. It provides a high-level view of the XSLT to aid in maintenance and new development of the conversion process. It will be used for "technical" documentation of XSL utilities so that they may be used across NCBI and released to the public. The program will also be used to document journal-specific assumptions and comments on source content sent to PubMed Central. This is essentially a "political" documentation of the content conversions.

2. Background

2.1. PubMed Central and Data Conversion Process

As a digital archive, PubMed Central receives various forms of the full text of article from the publishers. PubMed Central workflow involves converting the text files, images, PDF, and supplementary files from the publisher and storing them in PubMed Central Public Access Database, before releasing to the public.

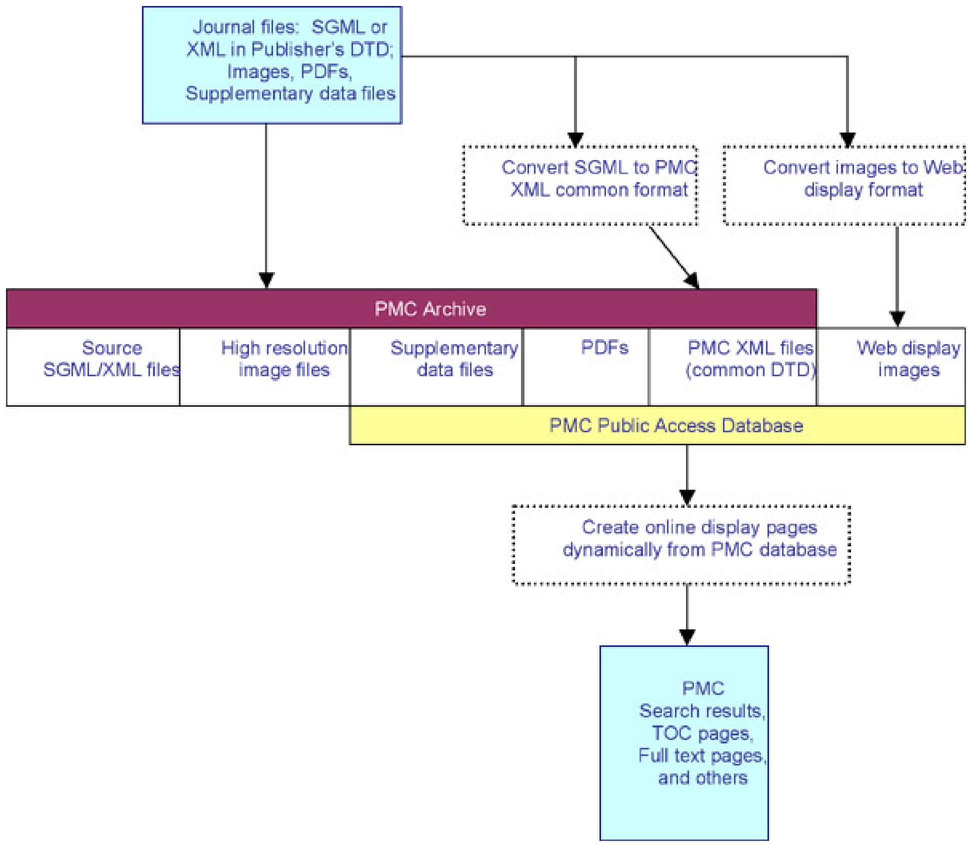

Journal files: SGML or XML in Publisher's DTD; Images, PDFs, Supplementary data files
Convert SGML to PMC XML common format
Convert images to Web display format
PMC Archive
Source SGML/XML files
High resolution image files
Supplementary data files
PDFs
PMC XML files (common DTD)
Web display images
PMC Public Access Database
Create online display pages dynamically from PMC database
PMC Search results, TOC pages, Full text pages, and others

Figure 1. PubMed Central Workflow

The data conversion process in PubMed Central involves converting the data format of the publishers to a data format defined by PubMed Central, JAIDTD (Journal Archiving and Interchange Document Type Definition) [http://dtd.nlm.nih.gov/]. Essentially, it's normalizing the publisher's format into PubMed Central DTD format. XSLT is the programming language that facilitates the conversion process.

Publishers' Data Format
XSL Transform
PubMed Central's Data Format JAIDTD

Figure 2. Data Conversion Process

Converting to a common DTD provides tremendous efficiencies. The complexities of handling diverse DTDs are dealt with once, when a file enters the archive. From there on, the uniform data format means simpler software for managing and retrieving data from the archive, as well as easier redistribution of the data to other archives *(Sequeira, 2003).*

NCBI recommends JAIDTD for standard data exchange for all online articles. If widely adopted, the JAIDTD would considerably streamline the process of archiving e-journals *(Peek, 2003)*. However, before the JAIDTD becomes the de-facto standard to publish online articles, the publishers will continue to supply various forms of XML or SGML to PubMed Central.

2.2. Digital Preservation

One of the key aspects of digital preservation has always been selection: which elements are preserved and why? These issues are well understood for prints on paper, they are still unclear for electronic resources.

Another key aspect of preservation has been format, and storage. What kind of file format should we preserve? How should we store it for long term access?

Before digital archives, traditional librarians and archivists make critical decisions on element selection, format and storage management. At PubMed Central, it's the developers at content conversion team from PubMed Central. They are responsible for data conversion process. They make decisions on element to element mapping, and decisions on long-term format, and storage management.

XSL Transform Self-Documentation is a program that will give a high-level overview of the data conversion process. In turn, these decision making processes can be better managed!

3. Project Goal

The project is to define a format and develop a system to document these decisions. It is to ensure quality and management of the data conversion process for PubMed Central. In addition, it provides technical documentation for shared utilities across NCBI.

In doing so, the information lifecycle can be well-understood by everyone involved. It ensures the process follows the guideline of electronic archival practices.

4. Technologies Involved

4.1. Definition of the Concepts

SGML (Standard Generalized Markup Language) is the original meta-language in which one can define markup languages for documents.

HTML (HyperText Markup Language) is a language that was defined using SGML: a set of tags to control formatting and behavior in online documents.

XML (Extensible Markup Language) is a simplified subset of SGML, capable of describing many different kinds of data. Its primary purpose is to facilitate the sharing of data across different systems, particularly systems connected via the Internet.

DTD (Document Type Definition) is a set of declarations that used to describe content in an describe content in an SGML, XML or HTML document, where each tag is allowed, and which tags can appear within other tags.

XSL (Extensible Stylesheet Language) is a family of languages which allows one to describe how files encoded in the XML standard are to be formatted or transformed. The family has three parts:

- XSLT (XSL Transformations): an XML language for transforming XML documents from one syntax to another.
- XSL-FO (XSL Formatting Objects): an XML language for specifying the visual formatting of an XML document.
- XPath (XML Path Language): a non-XML language used by XSLT, and XLink, to access or refer to parts of an XML document.

4.2. XML as an Archival Format

For long term digital preservation, XML is the archival format used in PubMed Central. The main advantage of the XML data format for long-term preservation is that it separates content from format, it's standard-based, and uses self-describing data *(Stewart, n.d.)*. Since XML is a standard maintained by the influential World Wide Web Consortium (W3C) [http://www.w3.org/], it guarantees future development and maintenance of XML. XML is also an excellent storage device for structurally rigorous document-centric information, such as biomedical journals.

Because of its self-describing features, the interpretation and rendering of XML formatted data is clear both for humans and computers. XML preserves the structure of an article by explicitly identifying individual elements, such as the article sections and section heads, or the journal, volume, issue and other details for each bibliographic reference. This explicit definition supports both accurate reproduction and reuse of content.

The best way to ensure the durability of an electronic archive is to use it constantly. Because PMC creates its online displays directly from the archival files, readers are confirming and validating the quality of the archival files every time they retrieve an article. This adds another level of assurance to NLM's own testing and quality control procedures *(Sequeira, 2003)*..

4.3. XSL Transform

XSLT (XSL Transform) is a declarative, XML-based document transformation language. It is a unique language that has a combined power of formatting style, programming, and query. It formats style like CSS, cascading style sheet; it processes data like Perl; it navigates through the hierarchical XML data structure like SQL. It can transform text, suppress content, duplicate, sort, and format text.

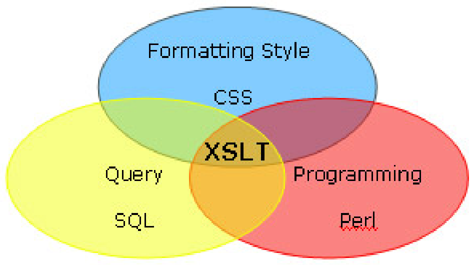


Figure 3. The world of XSLT

There are generally two functions of XSLT. The first function is a structural transformation, in which the data is converted from the structure of the incoming XML document to a structure that reflects the desired output. The second function is formatting, in which the new structure is output in the required format such as HTML or PDF *(Kay, 1999)*.

By separating content from style, XSLT enables reuse of fragment of data, produces multiple output formats, and styles tailored to user's preference. In the data conversion applications, XSLT can be used for extracting the data selectively, reordering it, turning attributes into elements or vice versa, or any number of similar tasks. It can also be used simply for validating the data *(Kay, 1999)*.

5. Methodology

5.1. Evaluation of Existing Tools

There is a limited number of existing self-documentation tools available. We examined six different methods. Some of them are open source projects that in development; some are methodologies other people are exploring. In addition to these systems, we looked at different formats of self-documentation, such as RDF, embedded XML, and structured comments.

1. XSLT Documentation Tool (Tool Kit included)
    - A full fledged XSLT documentation tool, embedded XSLT documentation code, using doc namespace, creates production XSL, and documentation XML.
    - The generation of the production stylesheets and documentation files can be automated with the help Apache Ant.
2. Document Your XSLT
    - Suggested doc tags for embedding structured documentation code. The stylesheet first needs to define the doc namespace by including the msxml:doc attribute on the stylesheet.
    - The doc namespace can be used to strip off the documentation if needed. Two versions of stylesheets can be kept – documented, and stripped undocumented.
    - Ideally, the doc namespace should be modeled using the Resource Description Framework (RDF) schema A full fledged XSLT documentation tool, embedded XSLT documentation code, using doc namespace, creates production XSL, and documentation XML, making it compliant with other XML specifications.
3. XSLT documentation generation XSLT (Tool Kit included)
    - Embedded comments
    - The comments, parameters, templates can be extracted from the MSXML parser.
4. Documenting style sheets using RDF
    - Resource Description Framework (RDF) is a standard mechanism provided by the W3C for just this purpose: describing metadata related to Web resources. de, using doc namespace, creates production XSL, and documentation XML.
    - Code comments are metadata, and RDF is an expressive and well-supported system that makes it easier to put the commentary to solid use.
5. XSLTdoc (Tool Kit included)
    - Embedded documentation - structured XSLT code
    - Tool requires Java and Saxon 8 jar class to run
6. pyXSLdoc (Tool Kit included)
    - Embedded documentation code, special characters, ala Javadoc
    - Tool requires python to run

From surveying these six methods, we find there are generally two types of documentation (user documentation and diagnostic documentation).

- User documentation describes the methods and parameters of an object, or the description of how a given class works *(Cagle, 2001)*.
- Diagnostic documentation, on the other hand, is intended to either explain why a given routine or variable is used. It's useful for debugging an application if it doesn't work properly *(Cagle, 2001)*.

5.2. Design & development decisions

After examining all the existing tools and formats, we decided to develop an in house system. The existing tools and formats are not sufficient enough to cover all of our needs.

We hope this new system will give us flexibility to:

- Set different Levels of Documentation
- Use Doc namespace to extract comments
- Document "political" comments
- Track Journal/Publication specific comments
- Most importantly, it gives us flexibility to grow the documentation, and build customized look and feel.

5.3. Documentation format

To make it easier for the developers to leave comments in the code, we decided to make the program flexible to take in 2 different documentation formats.

1. Using Documentation elements

   Advantages: The elements are written in XML. There is a DTD for the documentation elements.

   Disadvantages: More codes to write for the developers.

2. Using XML comments

   Advantages: It minimizes the code developers have to write to self-document.

   Disadvantages: It's less structured.

To summarize, we decided to allow the developers to use both formats. The first format is the recommended approach because it's formal and more structured. The 2nd format is more practical, involves very little code. The developers will most likely to adopt this approach.

The program will be able to take both formats. It converts the XML comments into a documentation element before further processing.

5.3.1. Documentation Elements

The documentation elements are written in XML. Because XSLT is expressed in XML too, it is necessary to define a new namespace to enable a XSLT processor to distinguish between documentation and source code. The URI for this namespace is http://www.nlm.nih.gov. This namespace must be declared in all stylesheets. The documentation element is the child element of the XSL element it's describing. There is a DTD description for the expected structure of the documentation elements.

```
DTD describes the document structure.

<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT doc:comment (doc:description, doc:daterequested, doc:requestedby,
doc:scope)>
<!ATTLIST doc:comment
author CDATA #REQUIRED
createddate CDATA #REQUIRED
xmlns:doc CDATA #IMPLIED
>
<!ELEMENT doc:daterequested (#PCDATA)>
<!ELEMENT doc:description (doc:para+)>
<!ATTLIST doc:description
level CDATA #IMPLIED
>
<!ELEMENT doc:journal (#PCDATA)>
<!ELEMENT doc:para (#PCDATA)>
<!ELEMENT doc:pub (#PCDATA)>
<!ELEMENT doc:requestedby (#PCDATA)>
<!ELEMENT doc:scope (doc:journal | doc:pub)+>
<!ELEMENT root (doc:comment+)>
```

Figure 4. Doc:comment DTD

Namespaces play an important role in XSLT. Their purpose is to allow you to mix tags from two different vocabularies in the same XML document. Since all documentation elements are in a separate namespace, it is easy to remove all documentation nodes when creating production XSL.

Namespaces are identified by a Unique Resource Identifier (URI). In this case, we add an URI to the namespaces of doc:comment to distinguish the self-documentation from the rest of the XSL elements.

```
<doc:comment xmlns="http://www.nlm.nih.gov">
```

Figure 5. Documentation Namespaces

Example 1. The documentation element is embedded inside a template.

```
   <xsl:template match="TOCSection" mode="pnas-series-title">
     <?nodepath node?>

     <doc:comment author="DH" createddate="3/30/05"
xmlns="http://www.nlm.nih.gov">
        <description>
           <para>Process TOCSection in a special mode to create series titles.
 This element
            becomes a series title if its content is "INAUGURAL ARTICLE". This
 is the only
           time when a series title is created. However, additional tests maybe
 added at a
            later time.</para>
        </description>
        <scope>
           <journal>pnas</journal>
        </scope>
     </doc:comment>

     <xsl:variable name="content">
        <xsl:call-template name="capitalize">
           <xsl:with-param name="str" select="node()"/>
        </xsl:call-template>
     </xsl:variable>

     <xsl:if test="$content = 'INAUGURAL ARTICLE'">
        <series-title>
           <xsl:apply-templates/>
        </series-title>
     </xsl:if>
  </xsl:template>
```

5.3.2. XML Comments

XML comments must proceed to the element it's describing. The comments can give the pre or post conditions of the XSL elements as well as name or explain any algorithm used. The special tagged symbols are used to declare the comments.

Re-format page sizes

```
#description: repeatable not required.
#level: only once not required.
#author: only once not required.
#datecreated: only once not required.
#daterequested: only once not required
#requestedby: only once not required.
#journal: repeatable not required.
#publication: repeatable not required.
```

Figure 6. Tag Convention

Example 2. The XML comment precedes the template it’s describing.

```
<!--
#description Process TOCSection in a special mode to create series titles.
This element becomes a series title if its content is "INAUGURAL ARTICLE".
This is the only time when a series title is created. However, additional
tests maybe added at a later time.
#author DH
#datecreated 3/30/05
#journal pnas
-->

<xsl:template match="TOCSection" mode="pnas-series-title">
     <?nodepath node?>

     <xsl:variable name="content">
        <xsl:call-template name="capitalize">
           <xsl:with-param name="str" select="node()"/>
        </xsl:call-template>
     </xsl:variable>

     <xsl:if test="$content = 'INAUGURAL ARTICLE'">
        <series-title>
           <xsl:apply-templates/>
        </series-title>
     </xsl:if>
  </xsl:template>
```

5.4. Documentation System

There are four steps involved processing XSL document in order to convert it into the desired output.

1. The first step is to transform unstructured comments into a documentation element. It conforms to the DTD we developed.
2. The second step is a structural transformation. In this step, the XSL elements and documentation XML are merged into a single XML documentation.
3. The third step is formatting. The documentation XML is formatted into HTML display.
4. Lastly, all comments are striped from XSL to produce a production XSL, free of documentation codes. This production XSL is what the data conversion process will be using.

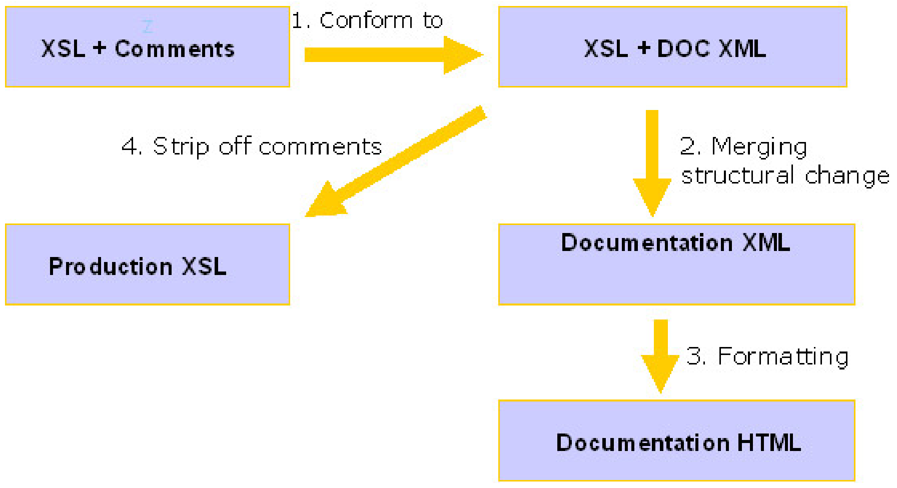


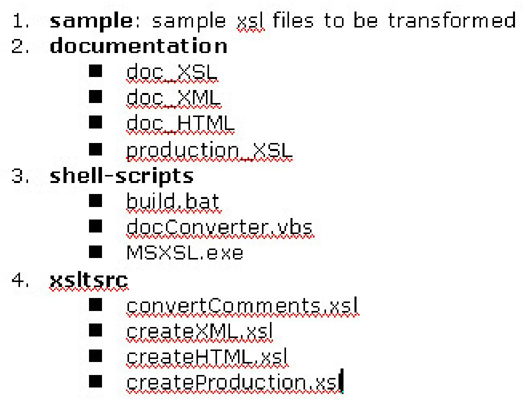
Figure 7. Documentation System

5.5. Assembling of the Tool Kit

A toolkit is assembled to test the format and the system we developed. Since this is only a prototype of what we envisioned in PubMed Central, the final system will be somewhat different, depending on the implementation of NCBI developers.

This toolkit is a standalone toolkit; it can be mounted on any Windows system. Because the program is written in XSLT, it's also platform independent. The source code may run against any operating systems with a XSL parser. In this case, we are using MSXSL parser from Microsoft; it's the same parser in every Internet Explorer Browser.

The XSLs are first placed in sample folder to be transformed. The generation of the production stylehsheets and documentation files are automated with the help of MSXML. Build.bat activates the docConverter.vbs which runs MSXML.EXE. The outputs are stored in the documentation folder.



1. **sample**: sample xsl files to be transformed
2. **documentation**
   - doc_XSL
   - doc_XML
   - doc_HTML
   - production_XSL
3. **shell-scripts**
   - build.bat
   - docConverter.vbs
   - MSXSL.exe
4. **xsltsrc**
   - convertComments.xsl
   - createXML.xsl
   - createHTML.xsl
   - createProduction.xsl

Figure 8. Folder Structure

To process, simply follow these steps:

- place xsl files in sample folder
- run Build.bat (shell-scripts folder)
- results in the documentation folder

Example 3. Shell-scripts Explained

```
docConverter.vbs takes in five parameters:
Source Folder, Results Folder, Source Format, Results Format, Engine

Example:

* convert free text comments into doc:comment ---
docConverter.vbs ..\sample ..\documentation\doc_XSL\ .xsl .xsl
convertComments.xsl

* convert xsl and doc:comment into documentation XML ---
docConverter.vbs ..\documentation\doc_XSL\ ..\documentation\doc_XML\ .xsl .xml
 createXML.xsl

* strip off doc:comments and create production XSL ---
docConverter.vbs ..\documentation\doc_XSL\ ..\documentation\production_XSL\
.xsl .xsl createProduction.xsl

* convert documentation XML into documentation HTML ---
docConverter.vbs ..\documentation\doc_XML\ ..\documentation\doc_HTML\ .xml
.html createHTML.xsl
```

6. Future works and implications

Since this is a prototype to test different methods of self-documentation, the next step for this project is the implementation phase. This will lie in the hands of NCBI developers. They will use this prototype and implement it to the existing system, and the developers will have to adjust the way they keep their documentation. Potentially, they could extend the DTD and add new elements to the self-documentation.

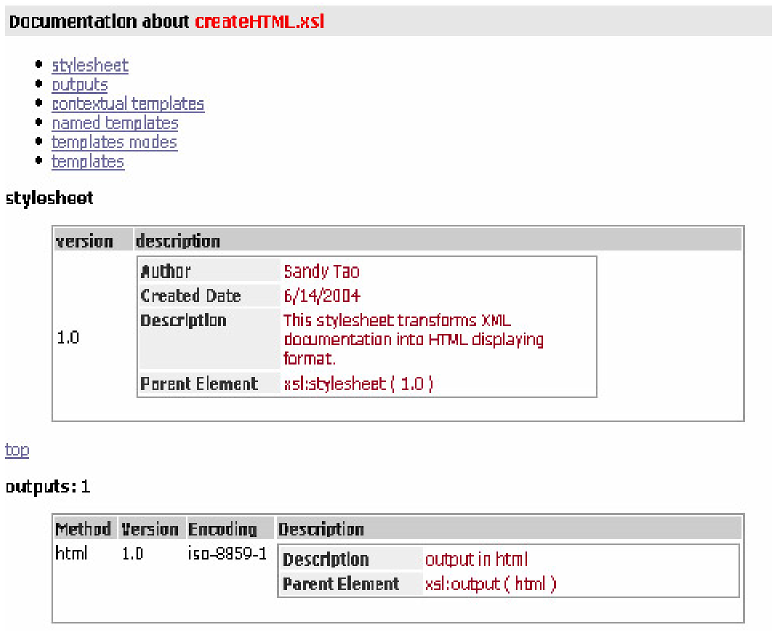
6.1. PubMed Central

With this XSL transform self-documentation tool, now the editorial comments can be tracked. Utility functions are better documented. It provides the developers with a high-level view of the data conversion process. Thus, the process can be better managed and maintained.

6.2. Other XSL programs

Potentially, this tool can be shared among NCBI for other self-documentation purposes. It can also be released to the public for other XSL programs to use.

Because this tool is written in XSLT, it actually documents itself! The following example is createHTML, the last step of the transformation. The source code runs through the toolkit and produces this self documentation.


Documentation about createHTML.xsl

- stylesheet
- outputs
- contextual templates
- named templates
- templates modes
- templates

stylesheet

version	description	
1.0	Author	Sandy Tao
	Created Date	6/14/2004
	Description	This stylesheet transforms XML documentation into HTML displaying format.
	Parent Element	xsl:stylesheet (1.0)

top

outputs: 1

Method	Version	Encoding	Description	
html	1.0	iso-8859-1	Description	output in html
			Parent Element	xsl:output (html)

Figure 9. Self-document createHTML.xsl

7. Notes

XML is becoming a standard Data Format for Digital Preservation. It is the archival format used in PubMed Central. The relevance of the standardized XML data format lies in its proclaimed non-proprietary, self-describing features. Storing digital objects as XML files is recognized for both long-term storage and access to the data they represent *(van Nispen, et al, 2005)*.

Archiving and Interchange DTD from NCBI is now slowly becoming a de-facto standard among the publishers who supply online data. Potentially, this data conversion process will not be needed as frequently, because the data format is becoming standardized.

We should note that XSLT is more than data conversion. It's also transforming text for data presentation layer. XSL transform self-documentation tool can also be used to document these transformations.

8. Discussion

Before digital archives, traditional librarians and archivists make critical decisions on element selection, format and storage management. Today, it's the programmers at NCBI, a division of National Library of Medicine. They are responsible for data conversion process. They make decisions on element to element mapping, decisions on storage management. They made decisions on preserving content over preserving format.

For long term preservation of digital contents, this information lifecycle needs to be documented and well-understood to ensure it follows the guideline of electronic archival practice. We must understand that digital preservation is not just a technical process, but an interactive and social process that involves many stakeholders.

Digital information is a complex network of hardware, software and media devices. Without a high level management tool, the information becomes very vulnerable to degradation; the content may become inaccessible as software and hardware change. It's essential to take action early in the life-cycle to make it simpler and less expensive to maintain in the future.

A well-understood process is to establish a standard, suitable benchmark, and evaluation procedures for accessing the outcome *(Lavoie & Dempsey, 2004)*. The self-documentation is a mechanism to track changes, and describes the processes, and potentially used to evaluate the process.

The Library has a long-term commitment to preserve the digital contents. These self-documentations will help the library to make critical decisions and manage these decisions. It is important these decision makers become responsible to secure the long-term persistence of the digital contents.

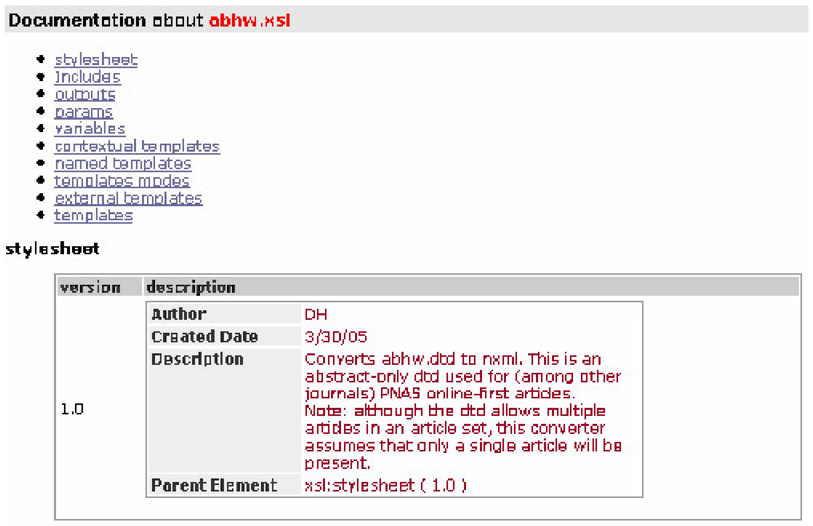
A. Source Code

The Source code is now checked into SourceForge. It is available from this url: [http://sourceforge.net/projects/xslt-selfdoc]

B. Sample Documentation

Example B.1. Highwire abstract - overview

This is an output of self-documentation HTML generated from Highwire abstract. The doc system summarizes various components of a stylesheet: such as include files, output files, global parameters, variables, different type of templates, and external templates. Under stylesheet, the documentation describes the stylesheet is to convert highwire abstract dtd to nxml.



Documentation about abhw.xsl

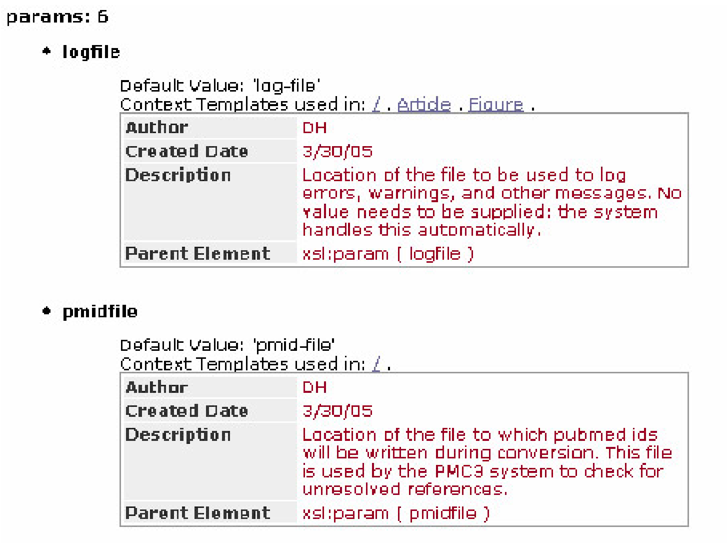
- stylesheet
- includes
- outputs
- params
- variables
- contextual templates
- named templates
- templates modes
- external templates
- templates

stylesheet

version	description
1.0	Author: DH Created Date: 3/30/05 Description: Converts abhw.dtd to nxml. This is an abstract-only dtd used for (among other journals) PNAS online-first articles. Note: although the dtd allows multiple articles in an article set, this converter assumes that only a single article will be present. Parent Element: xsl:stylesheet (1.0)

Example B.2. Highwire abstract - global parameters

Highwire abstract contains 6 global parameters. The first two are shown here. They contain default value, where they were used, and a description on how they were used.



params: 6

- **logfile**

Default Value: 'log-file'
Context Templates used in: / , Article , Figure .

Author	DH
Created Date	3/30/05
Description	Location of the file to be used to log errors, warnings, and other messages. No value needs to be supplied: the system handles this automatically.
Parent Element	xsl:param (logfile)

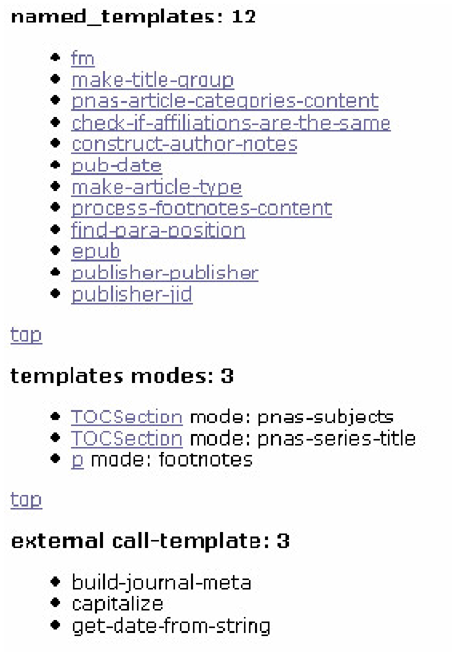
- **pmidfile**

Default Value: 'pmid-file'
Context Templates used in: / .

Author	DH
Created Date	3/30/05
Description	Location of the file to which pubmed ids will be written during conversion. This file is used by the PMC3 system to check for unresolved references.
Parent Element	xsl:param (pmidfile)

Example B.3. Highwire abstract - templates

Highwire abstract contains 12 name templates, 3 template modes, and 3 external call-templates.



named_templates: 12

- fm
- make-title-group
- pnas-article-categories-content
- check-if-affiliations-are-the-same
- construct-author-notes
- pub-date
- make-article-type
- process-footnotes-content
- find-para-position
- epub
- publisher-publisher
- publisher-jid

top

templates modes: 3

- TOCSection mode: pnas-subjects
- TOCSection mode: pnas-series-title
- p mode: footnotes

top

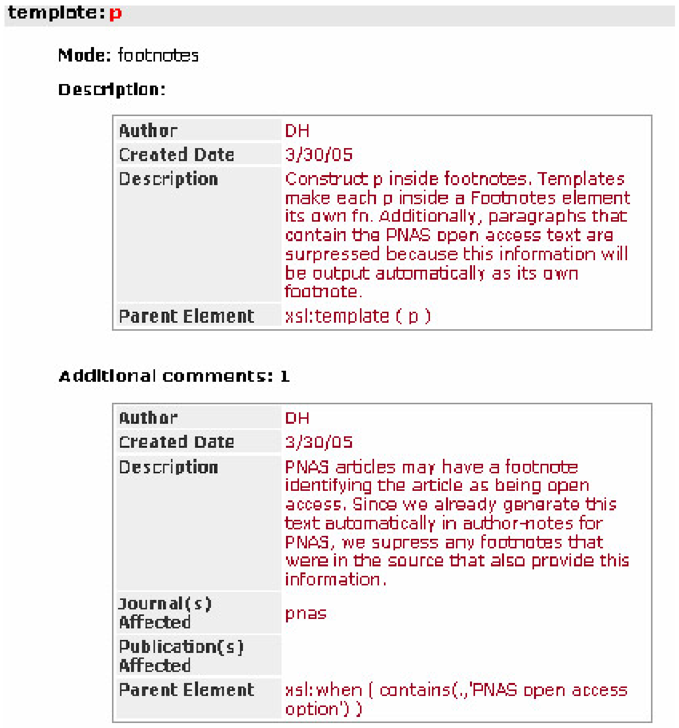
external call-template: 3

- build-journal-meta
- capitalize
- get-date-from-string

Example B.4. Highwire abstract - template p

This template p has 2 comments in it. One explains how this template behaves under normal circumstance; it will proceed to construct a paragraph tag inside footnote.

However, if the text contains “PNAS open access”, this template will suppress the footnotes because the text is already generated in author notes. This is a journal specific comment. In this case, it’s suppressing redundant information.



template: p

Mode: footnotes

Description:

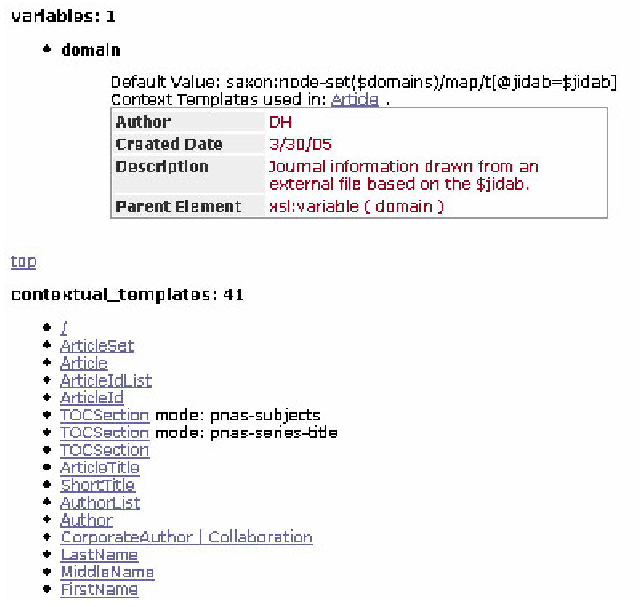
Author	DH
Created Date	3/30/05
Description	Construct p inside footnotes. Templates make each p inside a Footnotes element its own fn. Additionally, paragraphs that contain the PNAS open access text are surpressed because this information will be output automatically as its own footnote.
Parent Element	xsl:template (p)

Additional comments: 1

Author	DH
Created Date	3/30/05
Description	PNAS articles may have a footnote identifying the article as being open access. Since we already generate this text automatically in author-notes for PNAS, we supress any footnotes that were in the source that also provide this information.
Journal(s) Affected	pnas
Publication(s) Affected	
Parent Element	xsl:when (contains(.,'PNAS open access option'))

Example B.5. Highwire abstract - contextual templates

Highwire Abstract has 41 contextual templates. A contextual template contains elements such as article, articleId, FirstName, LastName, and Authors, etc. These templates are bibliographic elements that are described in the DTD.



variables: 1

- **domain**

Default Value: saxon:node-set($domains)/map/t[@jidab=$jidab]
Context Templates used in: Article ,

Author	DH
Created Date	3/30/05
Description	Journal information drawn from an external file based on the $jidab.
Parent Element	xsl:variable (domain)

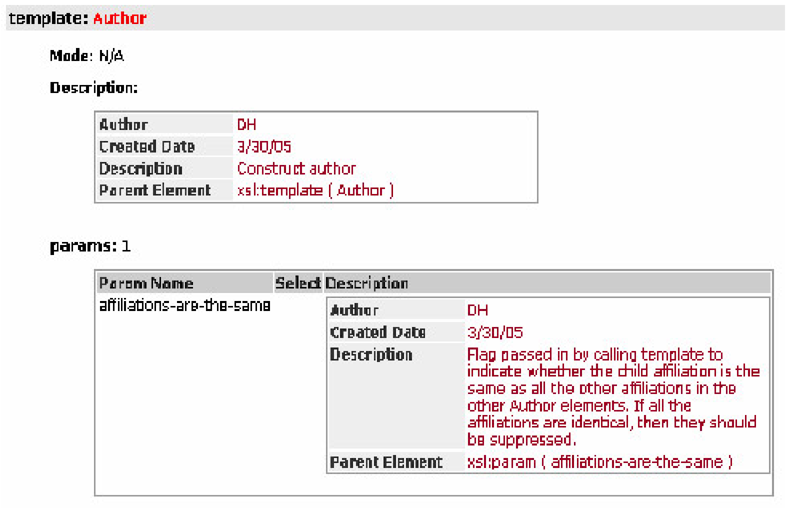
top

contextual_templates: 41

- /
- ArticleSet
- Article
- ArticleIdList
- ArticleId
- TOCSection mode: pnas-subjects
- TOCSection mode: pnas-series-title
- TOCSection
- ArticleTitle
- ShortTitle
- AuthorList
- Author
- CorporateAuthor | Collaboration
- LastName
- MiddleName
- FirstName

Example B.6. Highwire abstract - template Author

This author template contains a number of comments that are embedded in the code. On this page alone, you can see, the comments are in both the template level and on a parameter called affiliations are the same. It's important to understand the documentation may appear in many different places, and at various levels.



template: Author

Mode: N/A

Description:

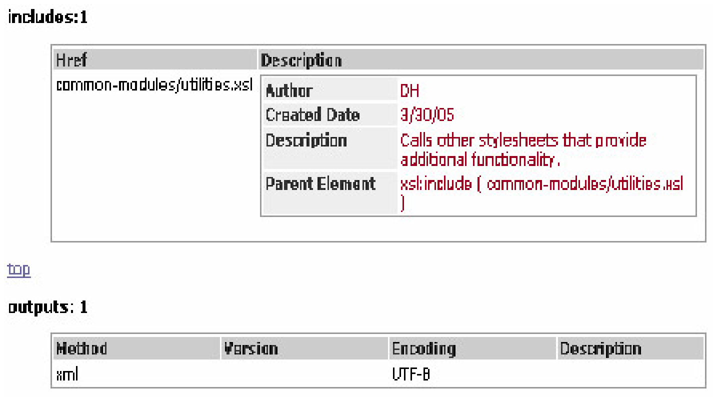
Author	DH
Created Date	3/30/05
Description	Construct author
Parent Element	xsl:template (Author)

params: 1

Param Name	Select	Description
affiliations-are-the-same		Author: DH; Created Date: 3/30/05; Description: Flag passed in by calling template to indicate whether the child affiliation is the same as all the other affiliations in the other Author elements. If all the affiliations are identical, then they should be suppressed.; Parent Element: xsl:param (affiliations-are-the-same)

Example B.7. Highwire abstract - includes

Highwire abstract has one include file called utilities. Utilities call other stylesheets to provide additional functionality.



includes:1

Href	Description
common-modules/utilities.xsl	Author: DH; Created Date: 3/30/05; Description: Calls other stylesheets that provide additional functionality.; Parent Element: xsl:include (common-modules/utilities.xsl)

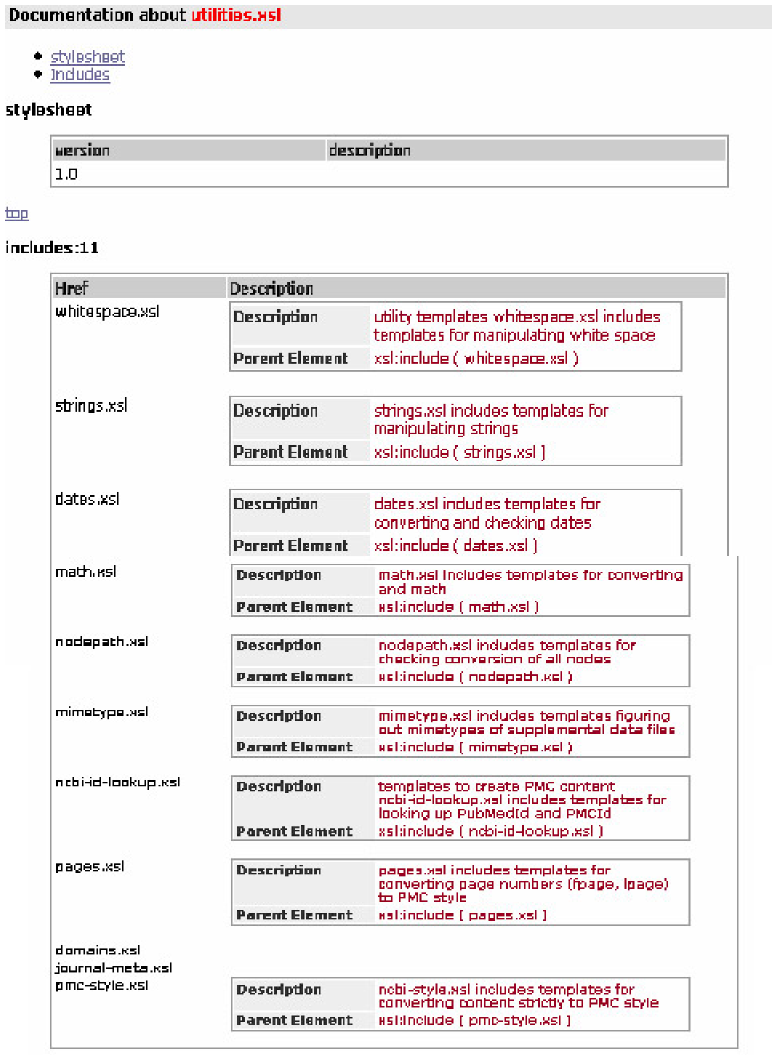
top

outputs: 1

Method	Version	Encoding	Description
xml		UTF-8	

Example B.8. Highwire abstract - utilities.xsl

This is the utilities stylesheet that calls on 11 other files for additional functionalities.



Documentation about utilities.xsl

- stylesheet
- includes

stylesheet

version	description
1.0	

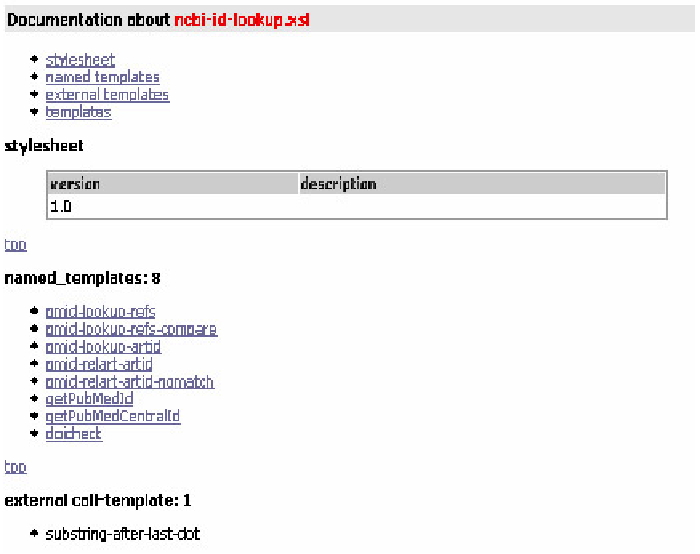
top

includes:11

Href	Description	
whitespace.xsl	**Description**	utility templates whitespace.xsl includes templates for manipulating white space
	Parent Element	xsl:include (whitespace.xsl)
strings.xsl	**Description**	strings.xsl includes templates for manipulating strings
	Parent Element	xsl:include (strings.xsl)
dates.xsl	**Description**	dates.xsl includes templates for converting and checking dates
	Parent Element	xsl:include (dates.xsl)
math.xsl	**Description**	math.xsl includes templates for converting and math
	Parent Element	xsl:include (math.xsl)
nodepath.xsl	**Description**	nodepath.xsl includes templates for checking conversion of all nodes
	Parent Element	xsl:include (nodepath.xsl)
mimetype.xsl	**Description**	mimetype.xsl includes templates figuring out mimetypes of supplemental data files
	Parent Element	xsl:include (mimetype.xsl)
ncbi-id-lookup.xsl	**Description**	templates to create PMC content ncbi-id-lookup.xsl includes templates for looking up PubMedId and PMCId
	Parent Element	xsl:include (ncbi-id-lookup.xsl)
pages.xsl	**Description**	pages.xsl includes templates for converting page numbers (fpage, lpage) to PMC style
	Parent Element	xsl:include (pages.xsl)
domains.xsl		
journal-meta.xsl		
pmc-style.xsl	**Description**	ncbi-style.xsl includes templates for converting content strictly to PMC style
	Parent Element	xsl:include (pmc-style.xsl)

Example B.9. Highwire abstract - ncbi-id-lookup.xsl

This is the ncbi-id-lookup stylesheet. It is what's used to look up PMIDs. As you can see from the quick demo, these sets of HTMLs can give you a very clear overview of the data conversion process, what templates are applied, and what templates are called. How some templates are suppressing data, and have an impact on a particular journal or publication. Often, a decision must be made to include or exclude certain elements. These are the editorial decisions the developers are called to make from time to time.



Documentation about ncbi-id-lookup.xsl

- stylesheet
- named templates
- external templates
- templates

stylesheet

version	description
1.0	

top

named_templates: 8

- pmid-lookup-refs
- pmid-lookup-refs-compare
- pmid-lookup-artid
- pmid-relart-artid
- pmid-relart-artid-nomatch
- getPubMedId
- getPubMedCentralId
- doicheck

top

external call-template: 1

- substring-after-last-dot

Acknowledgements

I would like to thank - Jeff Beck and Demian Hess, the tech guru at NCBI, for their involvement in this project. Also, I would like to thank Barbara Rapp, the coordinator for the NLM associate fellowship, for her guidance and encouragement in the fellowship year.

Bibliography

[Cagle] *Document Your XSLT* , K. Cagle, Feb 2001. Available at http://www.devx.com/getHelpOn/10MinuteSolution/20343/0/page/1.

[Kay] *XSLT Programmer's reference*, Kay, Wrox Press Limited , 1999. Available at http://www.topxml.com/xsl/articles/xslt_what_about/31290101.asp.

[Kielen] *XSLT Documentation Tool* , Kielen, 2004. Available at http://www.kielen.com/.

[Lavoie & Dempsy] *Thirteen Ways of Looking at...Digital Preservation* , Lavoie & Dempsy, D-Lib Magazine, 10:7/8 ,July/August, 2004. Available at http://www.dlib.org/dlib/july04/lavoie/07lavoie.html.

[Peek] *NLM Proposes New Journal Standards* , Peek , InfoToday,June, 2003. Available at http://www.infotoday.com/newsbreaks/nb030623-1.shtml.

[pyXSLdoc] *pyXSLdoc* , n.d. Available at http://cthedot.de/pyxsldoc/.

[RDF] *Documenting style sheets using RDF* ,n.d. Available at http://www-106.ibm.com/developerworks/xml/library/x-tipxrdf.html.

[Sequeira] *PubMed Central--Three Years Old and Growing Stronger* , Sequeira , ARL,228 : 5-9 ,June, 2003. Available at http://www.arl.org/newsltr/228/pubmed.html.

[Stewart] *What Is XML and Why Should I Care?* Stewart ,n.d . Available at http://www.topxml.com/xml/articles/what-isxml/.

[van Nispen] *The eXtensible Past* , van Nispen, et al. , D-Lib Magazine,11:2 ,Feb 2005. Available at http://www.dlib.org/dlib/february05/vannispen/02vannispen.html.

[XSLT] *XSLT documentation generation XSLT* ,n.d. Available at http://sourceforge.net/projects/xsltdocxslt.

[XSLTdoc] *XSLTdoc* , Jan 2005. Available at http://www.pnp-software.com/XSLTdoc/index.html.

Biography

Sandy **Tao**
NLM Associate Fellow
University of Washington, Health Sciences Libraries
Seattle
Washington
United States of America

Ms. Sandy Tao is a fellow from National Library of Medicine. She graduated from San Jose State University in 2004 with a MLIS degree. Her current interests are in information systems development, institutional repositories, open access issues, and environmental health. She is currently spending a year at the University of Washington.

Jeff **Beck**
Technical Information Specialist
National Center for Biotechnology Information [http://www.ncbi.nlm.nih.gov/]
National Library of Medicine, NIH
Bethesda
Maryland
United States of America

Mr. Beck works on the PubMed Central and NLM BookShelf projects at the National Library of Medicine managing the flow of journal and textbook content. He has been working with XML at NLM for 3 years. He has an editorial background, and his previous (relevant) experience includes print production and online production for journals and textbooks.

Demian **Hess**
Technical Information Specialist
National Center for Biotechnology Information [http://www.ncbi.nlm.nih.gov/]
National Library of Medicine, NIH
Bethesda
Maryland
United States of America

Mr. Hess works on the PubMed Central at the National Library of Medicine.