How to Publish XML Documents in HTML and PDF (2.1 legacy document)

Table of contents

Table of contents

Warning:
This document was copied as is from the Cocoon 2.1 documentation, but has not yet been fully reviewed or moved to its new home.

1. Overview

This How-To shows you how to publish XML documents in HTML and PDF using Cocoon. It requires no prior knowledge of Cocoon, XSLT or XSL-FO.

It has been updated for Cocoon 2.1, which does not require the use of the *mount* directory anymore.

2. Purpose

You will learn how to build a simple pipeline that converts XML documents on-the-fly to HTML or PDF using simple XSLT transforms. This is similar to the hello.html and hello.pdf samples of the standard Cocoon installation. However, this How-To teaches you how to build these mechanisms yourself. Thus, you will get a better feel of how Cocoon publishing really works.

3. Intended Audience

Beginning Cocoon users who want to learn how to publish HTML and/or PDF documents from XML data.

4. Prerequisites

Here's what you need:

- Cocoon must be running on your system. The steps below have been tested with Cocoon 2.1m3-dev, but they should work with any 2.1 version.
- This document assumes a standard installation where Cocoon is started by the *cocoon.sh* (or cocoon.bat) script and where http://localhost:8888/ points to the *Welcome to Apache Cocoon* page. If your installation runs on a different URL, you will have to adjust the URLs provided throughout this How-To as necessary.
- You must be able to create and edit XML files in the main directory of the Cocoon installation. When started from cocoon.sh, this directory is build/webapp under the directory that contains cocoon.sh.

You will not need a fancy XML editor for this How-To. Copying and pasting the sample code snippets into any text editor will do.Running "build clean" deletes everything under build/webapp, make sure to save your example files if you need to do a clean build.

5. Steps

Here's how to proceed.

5.1. 1. Create the work directory

Under build/webapp, create a new directory and name it html-pdf. All files used by this How-To will reside in this directory.

At this point, http://localhost:8888/html-pdf/ should display an error page saying *Resource not found*, indicating that the file *build/webapp/html-pdf/sitemap.xmap* was not found. This is normal, as the newly created directory does not yet contain the required sitemap file.

5.2. 2. Create the XML example documents

To keep it simple we will use two small XML files as our data sources. Later, you will probably use additional data sources like live XML feeds, databases, and others.

In the html-pdf directory, create the following two files, and name them exactly as shown.

Contents of file **pageOne.xml**:

<?xml version="1.0" encoding="iso-8859-1"?> <page> <title>This is the pageOne.xml example</title> <s1 title="Section one"> This is the text of section one </s1> </page>

Contents of file **pageTwo.xml**:

<?xml version="1.0" encoding="iso-8859-1"?> <page> <title>This is the pageTwo.xml example</title> <s1 title="Yes, it works"> Now you're hopefully seeing pageTwo in HTML or PDF </s1> </page> Be careful about the use of lower/uppercase in filenames if you're working on a Unix or Linux system. On such systems, thisFile.xml is not the same as Thisfile.xml.To avoid any errors, use copy/paste when creating XML documents from examples on this page.Do not leave spaces at the start of XML files. The <?xml... processing instruction must be the first character in the file.

5.3. 3. Create the XSLT transform for HTML

The most common way of producing HTML in Cocoon is to use **XSLT transforms** to select and convert the appropriate elements of the input documents.

Copy the file shown below to the html-pdf directory alongside your XML documents, naming it **doc2html.xsl**

<?xml version="1.0" encoding="iso-8859-1"?> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"> <!-- generate HTML skeleton on root element --> <xsl:template match="/"> <html> <head> <title><xsl:apply-templates select="page/title"/></title> </head> <body> <xsl:apply-templates/> </body> </html> </xsl:template> <!-- story is used later by the Meerkat example --> <xsl:template match="p|story"> <xsl:apply-templates/> </xsl:template> <!-- convert sections to HTML headings --> <xsl:template match="s1"> <h1><xsl:apply-templates select="@title"/></h1> <xsl:apply-templates/> </xsl:template> </xsl:stylesheet> Basically what this does is generate an HTML skeleton and convert the input markup to HTML. We won't go into details here. Rather, our goal is to show you how the components of the publishing chain are combined.

5.4. 4. Create the sitemap

We now have documents to publish and an XSLT transform to convert them to our HTML output format. What's left is to connect them in a **processing pipeline**. Then, the **sitemap** can select the pipeline based on the details of the browser request.

To tell Cocoon how to process requests made to html-pdf, copy the following snippet to a file named **sitemap.xmap** in the html-pdf subdirectory.

<?xml version="1.0" encoding="iso-8859-1"?> <map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0"> <!-- define the Cocoon processing pipelines --> <map:pipelines> <map:pipeline> <!-- respond to *.html requests with our docs

processed by doc2html.xsl --> <map:match pattern="*.html"> <map:generate src="{1}.xml"/> <map:transform src="doc2html.xsl"/> <map:serialize type="html"/> </map:match> <!-- later, respond to *.pdf requests with our docs processed by doc2pdf.xsl --> <map:match pattern="*.pdf"> <map:generate src="{1}.xml"/> <map:transform src="doc2pdf.xsl"/> <map:serialize type="fo2pdf"/> </map:match> </map:pipeline> </map:pipelines> </map:sitemap> The important thing here is the first **map:match** element, which tells Cocoon how to process requests ending in *.html in this directory. Again, we won't go into details here, but that's where it happens.The above sitemap is already configured for PDF publishing. However, this capability is not fully functional at this time because we haven't created the required XSLT transform yet.

5.5. 5. Test the HTML publishing

At this point you should be able to display the results in HTML:

- http://localhost:8888/html-pdf/pageOne.html should display the first page with "Section one" in big letters.
- http://localhost:8888/html-pdf/pageTwo.html should display the second page with "Yes it works" in big letters.

If this doesn't work, you might want to double check the above steps first, and then look at the Cocoon logs (see the Cocoon wiki for information about the logs).To convince yourself that the HTML data is generated dynamically, you can try to edit the pageXXX.xml source documents (keeping them well-formed), and refresh the browser to see the effect of your changes.

5.6. 6. Create the XSLT transform for PDF

PDF documents are created via XSL-FO documents which are XML documents that use a specific page-description vocabulary. (See References below for more info). The actual conversion to PDF is done by the PdfSerializer which uses software from FOP, another Apache Software Foundation project.

To activate the PDF conversion, copy the code snippet shown below to the html-pdf directory along with your XML documents, and name it **doc2pdf.xsl**

<?xml version="1.0" encoding="iso-8859-1"?> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0" xmlns:fo="http://www.w3.org/1999/XSL/Format" > <!-- generate PDF page structure --> <xsl:template match="/"> <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"> <fo:layout-master-set> <fo:simple-page-master master-name="page" page-height="29.7cm" page-width="21cm" margin-top="1cm" margin-bottom="2cm" margin-left="2.5cm" margin-right="2.5cm" > <fo:region-before extent="3cm"/> <fo:region-body margin-top="3cm"/> <fo:region-after extent="1.5cm"/> </fo:simple-page-master> <fo:page-sequence-master master-name="all"> <fo:repeatable-page-master-alternatives> <fo:conditional-page-master-reference master-reference="page" page-position="first"/> </fo:repeatable-page-master-alternatives> </fo:page-sequence-master> </fo:layout-master-set> <fo:page-sequence master-reference="all"> <fo:flow flow-name="xsl-region-body"> <fo:block><xsl:apply-templates/></fo:block> </fo:flow> </fo:page-sequence> </fo:root> </xsl:template> <!-- process paragraphs --> <xsl:template match="p"> <fo:block><xsl:apply-templates/></fo:block> </xsl:template> <!-- convert sections to XSL-FO headings --> <xsl:template match="s1"> <fo:block font-size="24pt" color="red" font-weight="bold"> <xsl:apply-templates select="@title"/> </fo:block> <xsl:apply-templates/>

</xsl:template> </xsl:stylesheet> This file is already referenced by the sitemap we created, so no additional configuration is needed.

5.7. 5. Test the PDF publishing

At this point you should be able to display the results in PDF in addition to the existing HTML versions:

- http://localhost:8888/html-pdf/pageOne.pdf should display the first page with "Section one" in big red letters.
- http://localhost:8888/html-pdf/pageTwo.pdf should display the second page with "Yes it works" in big red letters.

6. Summary

I hope you're beginning to see that publishing PDF and HTML documents in Cocoon is not too complicated, once you know what goes where.

The nice thing is that all of our huge corpus of XML documents (actually, only two documents right now, but that's a start...) is processed by just two XSLT transforms, one for each target format.

If you need to change the appearance of the published documents, you have to change only these two XSLT transforms. There's no need to touch the source documents.

7. Tips

7.1. Tip 1: Dynamic XML data

Using dynamic XML as the data source is very easy because the Cocoon FileGenerator can read URLs as well.

If you add the map:match element shown in bold below **before** the existing map:match elements in your sitemap.xmap file, requesting http://localhost:8888/html-pdf/meerkat.html should display real-time news from Meerkat (assuming an Internet connection to Meerkat is available).

The news will be displayed in a very rough format. However, this can be improved by writing a specific XSLT transform for this Meerkat data and using it, instead of doc2html.xsl, in the meerkat.html pipeline.

... <map:pipeline> **<map:match pattern="meerkat.html"> <map:generate src="http://www.oreillynet.com/meerkat/?_fl=xml"/> <map:transform src="doc2html.xsl"/> <map:serialize type="html"/> </map:match>** <map:match pattern="*.html"> etc...

7.2. Tip 2: Two-step conversion

When you are generating multiple formats from a single data source, it is often a good idea to generate an intermediate **logical document** that describes the output in a format-neutral way.

This is obviously not needed in our simple example. If you're aiming for more complicated publishing tasks, then you might want to read about this "publishing pattern" in Martin Fowler's Two Step View article.

8. References

To go further, you will need to learn about the following technologies and tools.

- Learning Cocoon concepts will help you understand how the sitemap, generators, transformers, and serializers work.
- Learning about XSLT will enable you to write your own transforms to generate HTML, PDF or other formats from XML data. Information about XSL-FO is available at the same address.

9. Comments

Care to comment on this How-To? Got another tip? Help keep this How-To relevant by passing along any useful feedback to the author, Bertrand Delacrétaz.

1. Comments

add your comments