

XML Transformation: CMSXDB

This tutorial describes how XML transformations are affected in the Oracle XML DB, by peering into the concepts, design, and implementation aspects that OTN developers applied to the Oracle XML DB Content Management System (CMSXDB). OTN developers used Oracle® JDeveloper and the features within Oracle XML DB to build the application.

This tutorial assumes that you are familiar with the CMSXDB and have installed and configured the required software as described in the [About the Content Management System XMLDB](#)

Contents

1. [Concepts](#)
2. [Design](#)
3. [Required Software](#)
4. [Setup](#)
5. [Implementation](#)
6. [Resources](#)
7. [Feedback](#)

Concepts

Understanding the following concepts will help further your understanding of how OTN developers designed and implemented the XML transformation feature in the CMSXDB:

- [About XSL](#)
- [About XML Transformation](#)
- [About Apache FOP](#)

About XSL

The content of a site that is managed by the Content Management System (CMS) is stored as XML documents within the XML DB repository. Very broadly, XML can be written using any tags we want - meaning there are no pre-defined XML tags that one should use. This also means that the tags used in XML, although possessing a defined structure, are not understood for display by the browser. **eXtensible Stylesheet Language (XSL)**, is the technology that adds formatting to a structurally defined XML document, by providing ways to display XML semantics and map XML elements into a universally browser-acceptable formatting language such as HTML.

XSL is comprised of three parts: XSLT (**eXtended Stylesheet Language Transformations**), XPath and XSL-FO. XSLT is a language that is used for transforming XML documents into other XML documents, XPath is a language for defining parts of an XML document and XSL-FO (XSL Formatting Objects) is a vocabulary that is used for formatting XML documents, thereby turning the result of an XSL Transformation into a suitable output form for a reader or listener.

XSL is a standard that is recommended by the World Wide Web Consortium (W3C) - the first two parts of the language (XSLT and XPath) became a W3C recommendation in November 1999. The full XSL recommendation including XSL formatting became a W3C recommendation in October 2001. Oracle XML DB complies with the W3C XSL/XSLT recommendation by supporting XSLT transformations in the database. XSLT Transformations, in the Oracle XML DB can be performed using either the `XMLTransform` function or the XMLType datatype's `transform` method.

About XMLTransformation

An XSL stylesheet is used to transform XML into other formats such as simple text or legacy HTML. XSLT provides a robust implementation of a tree-oriented transformation that takes the tree representation of an XML document and compares the nodes in the tree to the instructions in the XSL stylesheet. When the instructions in the XSL stylesheet is matched with the nodes in the tree, the XSLT processor outputs a tree fragment. The resulting tree fragments are collated to form a complete output tree, which is then serialized to a format such as text, HTML or an XML file. We will illustrate the above XML Transformation process using a sample XML file and XSL stylesheet, and finally look at the output of our transformation.

Our sample XML file (otnsample.xml) looks like this:

```

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="sample.xsl"?>
<otnsamples>
  <sample id="sm001">
    <author>Doe, John</author>
    <title>CMS XML DB</title>
  </sample>
  <sample id="sm002">
    <author>Barbara, Doe</author>
    <title>Financial Brokerage System</title>
  </sample>
  <sample id="sm003">
    <author>Stephens, Jon</author>
    <title>Oracle9i Web Services Security</title>
  </sample>
</otnsamples>

```

Our sample XSL file (otnsample.xsl) looks like this:

```

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/otnsamples">
    <HTML>
      <BODY>
        <TABLE BORDER="2">
          <TR>
            <TH>Title</TH>
            <TH>Author</TH>
          </TR>
          <xsl:for-each select="sample">
            <TR>
              <TD><xsl:value-of select="title"/></TD>
              <TD><xsl:value-of select="author"/></TD>
            </TR>
          </xsl:for-each>
        </TABLE>
      </BODY>
    </HTML>
  </xsl:template>
</xsl:stylesheet>

```

On applying our sample XSL file to the sample XML file, we get the following output on our browser:

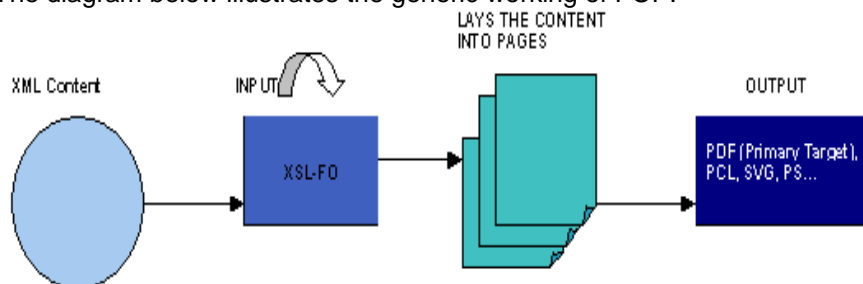
| Title | Author |
|--------------------------------|---------------|
| CMS XML DB | Doe, John |
| Financial Brokerage System | Barbara, Doe |
| Oracle9i Web Services Security | Stephens, Jon |

About Apache FOP

FOP (Formatting Object Processor) is a Java application driven by XSL-FO, that reads a Formatting Object (FO) tree and renders the resulting pages to a specified output. The primary target output for FOP is PDF, although it's worth noting that it currently supports the following formats too: PDF, PCL, PS, SVG, XML (area tree representation), Print, AWT, MIF and TXT.

One of the target output formats for the CMSXDB is PDF, and FOP is applied only when the PDF format is required. Note that the primary target output format of the CMSXDB is HTML.

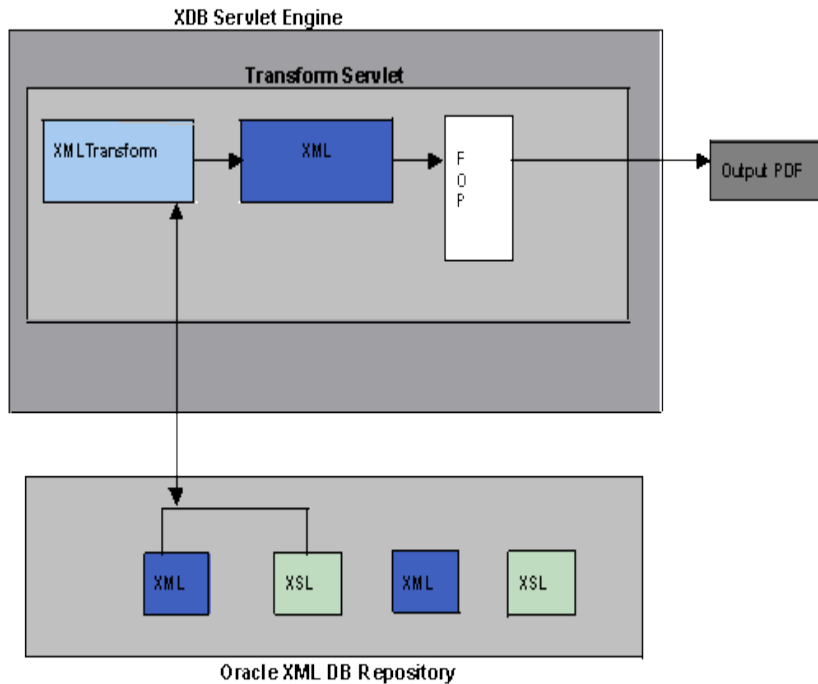
The diagram below illustrates the generic working of FOP:



Design

In the CMSXDB, when XML documents are required to be transformed and generated into their usable mode, the documents are fetched from the XDB and transformed by applying the specified XSL file. The transformed XML file generates an output that is written to the response, which is subsequently read by the browser that renders the page. It should be noted that the application server generates the file only when a file is externalized.

The diagram below illustrates the entire process of XML transformation that takes place within the CMSXDB:



When a file is required to be generated for preview or externalization, a **TransformServlet** that is deployed in the XDB servlet container is called with the resource name, XSL file and output content type. The servlet applies the XSL and generates the required content. It should be noted that the servlet in turn uses the `XMLTransform` database transformation function that applies the XSL and generates the output. It is also worth noting that using the `XMLTransform` function optimizes the transformation performance features such as memory usage, I/O operations, and network traffic as the transformation takes place close to data.

If the content type is PDF, Apache FOP is applied on the output that was generated as result of the database transformation.

Other aspects of the CMSXDB are covered in various lessons in this tutorial series.

Required Software

This tutorial presents some code examples. If you want to study them in context, download and install the [CMSXDB source code](#). If you also want to build and run the CMSXDB application, you will need the software listed in the [Required Software](#) section of About the CMS XML DB Application.

JDeveloper is Oracle's visual Java development tool and can be downloaded from [here](#).

Setup

If you use Oracle9i JDeveloper as your IDE, add the J2EE library to your project. For coding details, see the [Implementation](#) section.

To configure your system to build and run the Oracle CMSXDB sample application, see the [Setup](#) section of *About the Oracle CMSXDB application*.

Implementation

In this section we will look at how OTN developers implemented the XML Transformation features in the CMSXDB, and also further our conceptual understanding on the topic areas that we learnt in the [Concepts](#) section of this tutorial.

The XML Transformation feature is implemented in the TransformServlet that is embedded in the XDB servlet engine. OTN developers used the `XMLTransform` function to affect transformations in the CMSXDB. The `XMLTransform` function takes in an `XMLType` instance and an XSLT stylesheet. It applies the stylesheet to the XML document and returns a transformed XML instance as specified in the XSL Stylesheet. This XML instance can be anything from simple text to legacy HTML or XML instances using any other vocabulary. If the content type is PDF, then Apache FOP is applied to the output that is generated as a result of database transformation.

Here is the SQL query that is being called to do the transform:

```
/* SQL Query to transform resource by applying XSL */
private static final String transformsql =
    " SELECT form( xdbUriType( ? ).getXML( )," + " xdbUriType( ? ).getXML( ) )."
    getClobVal() " + " FROM DUAL ";
```

It is worth noting that we pass two arguments to the `XMLTransform` function - the first one being the `XdbUri` of the resource and the other being the `XdbUri` of the XSL to be applied. The output of the transformation is returned as a CLOB value.

After the database transformation is affected, if the content type has been specified as PDF, then Apache FOP is applied to the output. Listed below is the FOP code that generates PDF and writes the o/p directly to the response:

```
if(fopDriver == null) {

    fopDriver = new Driver( );
    System.out.println("fop driver loaded");

    // Initialize Logger for FOP
    Hierarchy hierarchy = Hierarchy.getDefaultHierarchy();
    fopLog = hierarchy.getLoggerFor("fop");
    fopLog.setPriority(Priority.WARN);
    System.out.println("loaded logger");

}

// Set parameters for rendering
fopDriver.setLogger(fopLog);
fopDriver.setInputSource( input );
fopDriver.setOutputStream( output );
fopDriver.setRenderer( this.getRenderer( outFormat ) );
System.out.println("render start");
try {
    // Actual rendering happens now
    fopDriver.run();
}
// handle error
```

Resources

This tutorial is part of a series, *Understanding Oracle XML DB*, based on the Content Management System (CMS) sample application. Following are links to resources that can help you understand and apply the concepts and techniques presented in the tutorials. See the [Required Software](#) section to obtain the CMS source code and related files.

| Resource | URL |
|----------------------------|---|
| Oracle XML DB Introduction | http://otn.oracle.com/tech/xml/xmlldb/htdocs/xmlldb_intro.html |
| JDeveloper Online Help | http://otn.oracle.com/jdeveloper903/help/ |
| Oracle XML DB | http://otn.oracle.com/tech/xml/xmlldb/content.html |
| Apache FOP | http://xml.apache.org/fop/ |
| OTN Sample Code | http://otn.oracle.com/sample_code/content.html |

Feedback

If you have questions or comments about this tutorial, you can:

- Post a message in the [OTN Sample Code discussion forum](#). OTN developers and other experts monitor the forum.
- Send email to the author. <mailto:Dilip.Thomas@oracle.com>

If you have suggestions or ideas for future tutorials, you can:

- Post a message in the [OTN Member Feedback forum](#).
- Send email to <mailto:Raghavan.Sarathy@oracle.com>