NEWS

Convert HTML content to PDF format

Support access to PDF files on your Webpages

By Nick Afshartous

JavaWorld | APR 10, 2006 1:00 AM PT

Making Web content available as PDF is one way to facilitate the dissemination of content. In some industries, providing access to print-formatted documents, such as employee benefit descriptions, is mandatory. The law actually dictates that summary plan descriptions (SPDs) be made available in print format even though the content may be provided online. Just printing the Webpage is not sufficient because the print format must include a table of contents with page number references.

To add such functionality to a Webpage, developers can convert the HTML content to PDF format; this article illustrates how. The method illustrated here to perform the conversion uses only open source components. Commercial products also support dynamic document generation. Adobe has the Document Server product line, for example; however, its cost is substantial. Using an open source solution mitigates the cost factor while adding source code transparency.

The conversion consists of three steps:

1. Convert the HTML to XHTML
2. Convert the XHTML document to XSL-FO (Extensible Stylesheet Language Formatting Objects) using an XSL stylesheet and an XSLT transformer
3. Pass the XSL-FO document to a formatter to generate the target PDF document

This article demonstrates how to perform the translations using the command line interfaces provided by the tools and then introduces a Java program that uses the DOM (Document Object Model) interfaces.

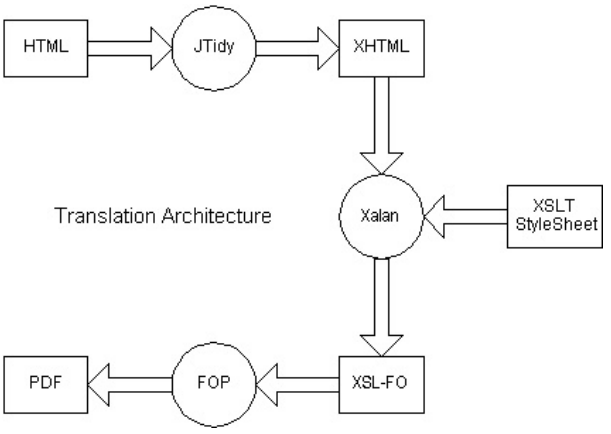
Component versions

The code in this article was tested with the following versions:

Component	Version
JDK	1.5_06
JTidy	r7-dev
Xalan-J	2.7
FOP	0.20.5

Using the command line interfaces

Each of the three steps consists of generating an output file from an input file. The inputs and outputs of the steps are shown in the figure below.



Translation

Using the three tools' command line interfaces allows for an easy way to get started. However, this approach is not suitable for a production-level system because of the temporary intermediate files that would be written to disk. This extra I/O would result in poor performance. Later in this article, the issue of temporary files becomes moot when the three tools are invoked by a Java program.

Step 1: HTML to XHTML

The first step is to translate the HTML file to a new XHTML file. Of course if the starting point for the conversion is already XHTML, then this step does not apply.

I used JTidy to perform the translation. JTidy is a Java port of the Tidy HTML parser. In the process of translating to XHTML, JTidy also adds missing close tags to create a well-formed XML document. I used the most recent version listed (r7-dev) on the SourceForge Website.

To run JTidy, use the following tidy.sh script:

```
#/bin/sh

java -classpath lib/Tidy.jar org.w3c.tidy.Tidy -asxml  >
```

This script sets the `CLASSPATH` variable and invokes JTidy. To run JTidy, the input file is passed as a command line argument. By default, the generated XHTML is directed to standard output. The `-modify` switch can also be used to overwrite the input file. The `-asxml` switch directs JTidy to output well-formed XML as opposed to HTML.

The script is invoked as:

```
  tidy.sh hello.html hello.xml
```

And the files hello.html (input) and hello.xml (output) are shown here:

```
<html>
<head>
  <title>Hello World
</head>
<body>
   <p> Hello World!
</body>
</html>

<!DOCTYPE html PUBLIC quot;-//W3C//DTD XHTML 1.0 Strict//EN"
    quot;http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<html xmlns=quot;http://www.w3.org/1999/xhtml">
<head>
<meta name=quot;generator" content="HTML Tidy, see www.w3.org" />
<title>Hello World
</head>
<body>
<p>Hello World!</p>
</body>
</html>
```

Note that the `<p>` tag has a closing `</p>` tag in the XML file added by JTidy.

Step 2: XHTML to XSL-FO

Next, the XHTML is transformed to XSL-FO, a language for specifying the print format of XML documents. To achieve the transformation, I apply an XSL stylesheet processed by an XSLT transformer (Apache Xalan). The stylesheet I used as a starting point is xhtml2fo.xsl, provided by Antenna House, a company that sells a commercial formatter for XSL-FO.

The xhtml2fo.xsl stylesheet specifies how each of the HTML tags is to be translated to the corresponding sequence of XSL-FO formatting commands. For instance, in case of an HTML `H2` tag, the translation is defined as:

```
<xsl:template match="html:h2">
   <fo:block xsl:use-attribute-sets="h2">
     <xsl:call-template name="process-common-attributes-and-children"/>
   </fo:block>
 </xsl:template>
```

The above XSLT template is invoked every time an `H2` tag is encountered in the HTML input stream. The `html:` prefix indicates that the `H2` tag is in the HTML namespace. The stylesheet's namespaces are specified as attributes to the top-level `xsl:stylesheet` directive. Looking at the top of the xhtml2fo.xsl file, we see three namespaces specified. One for each of the XSL, XSL-FO, and HTML languages:

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:fo="http://www.w3.org/1999/XSL/Format"
```

```
                    xmlns:html="http://www.w3.org/1999/xhtml">...
```

The second line of the template

```
    <fo:block xsl:use-attribute-sets="h2">
```

causes an `fo:block` tag to be output, with the `H2` attribute set to generate the attributes and values for the `fo:block` tag. An XSL-FO block is a region of text rendered based on the values specified for the block's attributes.

The attribute set for `H2` is defined in the stylesheet as:

```
    <xsl:attribute-set name="h2">
         <xsl:attribute name="start-indent">10mm
         <xsl:attribute name="end-indent">10mm
         <xsl:attribute name="space-before">1em
         <xsl:attribute name="space-after">0.5em
         <xsl:attribute name="font-size">x-large
         <xsl:attribute name="font-weight">bold
         <xsl:attribute name="color">black
    </xsl:attribute-set>
```

The `start-indent` and subsequent attributes above are used to specify the formatted appearance of an `H2` block. Using an attribute set makes it easy to change the appearance of all blocks in the PDF document that correspond to the same HTML input tag. Simply change the settings in the attribute set; the output of all translations that use the attribute set will change too.

The next directive in the translation calls the template named "process-common-attributes-and-children":

```
    <xsl:call-template name="process-common-attributes-and-children"/>
```

This template is defined in the stylesheet. Its purpose is to check for some common HTML attributes (i.e., lang, id, align, valign, style) and then generate the corresponding XSL-FO directives. To trigger the translation of any tags nested within the top-level `H2` tag, the

process-common-attributes-and-children template then calls:

```
   <xsl:apply-templates/>
```

Hence, if the input is

```
   <h2> Hello <em> there </em> </h2>
```

the `<xsl:apply-templates/>` inside the template for `H2` would trigger the invocation of the template that translates the `<em>` tag.

The output of translating an `H2` tag is:

```
   <fo:block start-indent="10mm" ...
        original H2 tag content
    </fo:block>
```

To apply the xhtml2f0.xsl, we invoke the Xalan transformer. The Unix script xalan.sh sets the `CLASSPATH` variable with the required jar files before calling Xalan:

```
#/bin/sh
export CLASSPATH='.;./lib/xalan.jar;./lib/xercesImpl.jar;./lib/xml-apis.jar;lib/serializer.jar'
java -classpath $CLASSPATH org.apache.xalan.xslt.Process -IN  -XSL xhtml2fo.xsl -OUT  -tt
```

Since Xalan requires an XML parser, the Apache Xerces parser and xml-api JARs are referenced in addition to xalan.jar. All of the jar files are bundled with the Xalan distribution.

To create an XSL-FO file by applying the stylesheet to the XHTML input, invoke the script:

```
   xalan.sh  hello.xml hello.fo
```

I like to use the trace option (`-tt`) with Xalan to display a trace of the templates that are

applied. The file hello.fo is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>

<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
    xmlns:html="http://www.w3.org/1999/xhtml"
    writing-mode="lr-tb"
    hyphenate="false"
    text-align="start"
    role="html:html">

  <fo:layout-master-set>
    <fo:simple-page-master page-width="auto" page-height="auto"
                           master-name="all-pages">
      <fo:region-body column-gap="12pt" column-count="1" margin-left="1in"
                      margin-bottom="1in" margin-right="1in" margin-top="1in"/>
      <fo:region-before display-align="before" extent="1in"
                        region-name="page-header"/>
      <fo:region-after display-align="after" extent="1in"
                       region-name="page-footer"/>
      <fo:region-start extent="1in"/>
      <fo:region-end extent="1in"/>
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-reference="all-pages">
    <fo:title>Hello World
    <fo:static-content flow-name="page-header">
      <fo:block font-size="small" text-align="center" space-before="0.5in"
                space-before.conditionality=;"retain">
        Hello World
      </fo:block>
    </fo:static-content>

    <fo:static-content flow-name="page-footer">
      <fo:block font-size="small" text-align="center" space-after="0.5in"
                space-after.conditionality=quot;retain">
        - <fo:page-number/> -
      </fo:block>
    </fo:static-content>

    <fo:flow flow-name="xsl-region-body">
      <fo:block role="html:body">
        <fo:block space-before="1em" space-after="1em" role="html:p">
```

```
        Hello World!
      </fo:block>
    </fo:block>
  </fo:flow>

 </fo:page-sequence>

</fo:root>
```

Step 3: XSL-FO to PDF

The third and final step is to pass the XSL-FO document to a formatter that can generate PDF. I used Apache FOP (Formatting Objects Processor). FOP partially implements the XSL-FO standard and best supports the PDF output format. There is nascent support for Postscript, and support for Microsoft's RTF (rich text format) is planned. The FOP distribution contains the shell script fop.sh/fop.bat that takes an XSL-FO file as input and generates the specified PDF output file.

The Unix script can be run, for example, by:

```
fop.sh hello.fo hello.pdf
```

The only prerequisite is to set the environment variable used by the script for the FOP home directory's path.

The file hello.pdf contains the output from FOP and is included in this article's source code, which can be downloaded from Resources.

Since FOP doesn't currently fully implement the XSL-FO standard, there are some limitations. The subset of the standard that is supported is described in detail in the compliance section of the FOP Website.

Java program

Utilizing the DOM APIs of the three tools used in the steps above, I'll now present a Java program that requires two command line arguments (HTML input file and stylesheet) and creates a corresponding PDF. No temporary files are created.

First the program creates an `InputStream` for the HTML file. The `InputStream` object is then passed to JTidy.

JTidy has method `parseDOM()`, which is called to obtain the output XHTML content as a `Document` object:

```
public static void main(String[] args) {

    // open file
    if (args.length != 2) {
        System.out.println("Usage: Html2Pdf htmlFile styleSheet");
        System.exit(1);
    }

    FileInputStream input = null;
    String htmlFileName = args[0];
    try {
        input = new FileInputStream(htmlFileName);
    }
    catch (java.io.FileNotFoundException e) {
        System.out.println("File not found: " + htmlFileName);
    }

        Tidy tidy = new Tidy();
    Document xmlDoc = tidy.parseDOM(input, null);
```

XML namespaces are not supported in the JTidy's DOM implementation; hence, the rules in the Antenna House stylesheet must be modified to use the default namespace. For example, instead of

```
  <xsl:template match="html:h2">
     <fo:block xsl:use-attribute-sets="h2">
       <xsl:call-template name="process-common-attributes-and-children"/>
     </fo:block>
   </xsl:template>
```

the rule is changed to

```
   <xsl:template match="h2">
      <fo:block xsl:use-attribute-sets="h2">
        <xsl:call-template name="process-common-attributes-and-children"/>
      </fo:block>
    </xsl:template>
```

This change must be applied to all templates in xhtml2f0.xsl because the `Document` object created by JTidy has `<html>` as the root tag, as opposed to:

```
<html xmlns=quot;http://www.w3.org/1999/xhtml">
```

The modified version of xhtml2fo.xsl is included in this article's source code.

Next, the method `xml2FO()` calls Xalan programmatically to apply the stylesheet to the DOM object created by JTidy:

```
   Document foDoc = xml2FO(xmlDoc, args[1]);
```

Method `xml2FO()` first calls `getTransformer()` to obtain a `Transformer` object for the specified stylesheet. The `Document` that represents the result of the transform is then returned:

Related: Java App Dev Java Language

1 | 2 | NEXT ›

Recommended

Eclipse, NetBeans, or IntelliJ? Choose your Java IDE

Android Studio for beginners: Code the app

Immutable empty collections and iterators

Open source Java projects: Docker Swarm

VIDEO/WEBCAST

SPONSORED

Building Cognitive IoT-Robotics-Mobile Messaging with Java, Watson and MobileFirst on Bluemix

Extending the value of your Java applications means more than just moving to the cloud. Business...

Popular on JavaWorld

Eclipse, NetBeans, or IntelliJ? Choose your Java IDE

Find out what to look for in a Java IDE and get tips for deciding which of the top three--Eclipse,...

Android Studio for beginners: Code the app

Open source Java projects: Docker Swarm

Newsletters

Stay up to date on the latest tutorials and Java community news posted on JavaWorld

Get our Enterprise Java newsletter

Enter your email address

GO

Lightning fast NoSQL with Spring Data Redis

Redis isn't your typical NoSQL data store, and that's exactly why it hits the sweet spot for certain...

Choosing your Java IDE

Find out what to look for in a Java IDE and get tips for deciding which of the top three--Eclipse,...

Popular Resources

Video/Webcast

Sponsored

Building Cognitive IoT-Robotics-Mobile Messaging with Java, Watson and MobileFirst on Bluemix

White Paper

IBM Bluemix - From Idea to Application

White Paper

Microservices Without the Hassle of Infrastructure

Video/Webcast

Sponsored

OpenWhisk Hello World Demonstration

Video/Webcast

Sponsored

PointSource Shortens Scrum Meetings by 50 Percent with IBM DevOps Services for Bluemix

Search Resources

Go

Featured Stories

Technology of the Year Award winners

InfoWorld editors and reviewers pick the year's best hardware, software, development tools, and cloud...

Google creates ‘crisis fund’ following US immigration ban

Tech giant, Google, has created a US$2 million crisis fund in response to US president Donald Trump’s...

Skills certification coming for Node.js developers

The Node.js Foundation is creating a Certified Developer program to help companies and developers gauge...

US tech industry says immigration order affects their operations

Apple, Microsoft, and Google are among U.S. tech industry leaders warning that Trump's immigration...

Explore the IDG Network