SINGLE-SOURCE INTERACTIVE AND PRINTED CONTENT PUBLISHING USING THE DOCBOOK XML STANDARD

D. MOLLOY
RINCE, School of Electronic Engineering, Dublin City University, IRELAND.
E-mail: Derek.Molloy@dcu.ie

When an author publishes e-Learning on-line course content for the first time, document format can be a difficult choice. If a print-friendly format is chosen (such as PDF, Word, PowerPoint) it can be difficult to embed interactive components, but if a web-friendly format is chosen (such as HTML) it can be difficult to provide a well-formatted paper version of the course content. DocBook is a collection of standards and tools for technical publishing, particularly suited to large quantities of highly structured content. The key difference between DocBook and other structured formats (including LaTeX) is that the style (bold, font size, italics etc.) is separated from the structured content, allowing a single source DocBook XML document to be rendered to any format with any defined style. This allows one source document to have many presentations, such as generic HTML, accessible HTML or PDF. The DocBook XML standard has been applied to a new engineering module in the University that has both traditional and on-line student attendance. This paper discusses the implementation environment, including additions such as interactive components (such as embedded examples and on-line self-assessment), personalized role-based publishing (beginner/advanced user), published document protection (personalized and encrypted PDF), accessible templates (for disabled users). Finally the limitations and technical difficulties with the application of DocBook to e-Learning course content publishing are discussed.

1 Introduction

When an academic first decides to place notes on a website, it can be difficult for them to choose a document format that allows the interactive features of the World-Wide Web (WWW) to be integrated with the traditional electronic media that may already exist for the module content. Commonly, the academic assumes that the only way to take advantage of the new features of the WWW is to convert/re-write the module notes using HTML editors, creating a whole new copy of the notes that must also be maintained. Several problems quickly emerge with the on-line HTML format if it is chosen as the sole format:

- Printing – The presentation format of the notes cannot be preserved. It can be difficult to control images, code-segments and formulae to remain within the page bounds. If the notes are spread across many hyper-linked HTML pages, it can be time consuming and difficult to print a version that is representative of the notes, unless the hyper-document is very carefully indexed.
- Quality – Formulae and vector-mapped images must be carefully converted to a compressed image format, such as .gif to allow generalised on-line viewing.
- Internet connectivity – students must usually remain on-line to access the content, as it can be difficult for the academic to provide a suitable off-line version.
- Protection of Content – HTML pages can simply be saved, or cut-and-pasted into HTML editors. It can be difficult to prevent copyrighted content from being abused.

While there are several difficulties with the on-line format, it is often the first content choice, as it provides a flexible format that allows on-line interactive content, such as server-side processing, Java applets, mailing-lists and hyper-linking to references, glossaries etc. If the traditional off-line format is used, then there are several problems that can also arise:

- If Microsoft Word/PowerPoint format is used, then the download files are large and problems can occur with versioning. The academic is also presenting the student with the original source material that can easily be modified and duplicated.
- If Adobe PDF format is used, file size is smaller and the document can be encrypted, so that the document can be protected from duplication.

Both of these formats allow well-formatted printing, however, the lack of interactive features is restrictive in presenting an on-line module.

In this paper, we present a system developed for presenting a post-graduate module in object-oriented programming at Dublin City University. This module has a combination of traditional lecture-based

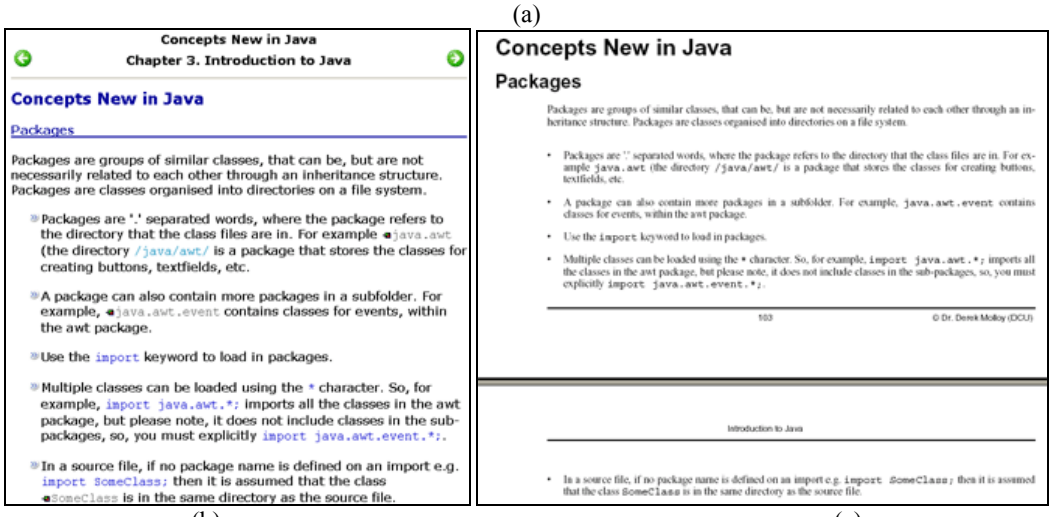
students (approx. 70), and on-line only/combined students (approx. 40). The developed system aims to allow single-source content to be converted into a format that suits off-line printing, on-line content to allow interactive features and a format to suit electronic presentation in lectures. Due to the dynamically changing nature of the content in this module, multiple source documents would be difficult to maintain.

2 Methods

The Oasis DocBook [1] format was examined for suitability for single-source interactive and printed content publishing. DocBook is a collection of standards and tools for technical publishing, particularly suited to large quantities of highly structured content. The key difference between DocBook and other structured formats (including LaTeX [2]) is that the style (bold, font size, italics etc.) is separated from the structured content, allowing a single source DocBook XML document to be rendered to any format with any defined style. This allows one source document to have many presentations, such as generic HTML, accessible HTML or PDF, as illustrated in Figure 1.

```
<sect1><title>Concepts New in Java</title>
<sect2><title>Packages</title>
<para>
Packages are groups of similar classes, that can be, but are not necessarily related to each other
through an inheritance structure. Packages are classes organised into directories on a file system.
<itemizedlist>
    <listitem><para>Packages are '.' separated words, where the package refers to the directory
      that the class files are in. For example <classname>java.awt</classname> (the directory
      <filename>/java/awt/</filename> is a package that stores the classes for creating buttons,
      textfields, etc.</para></listitem>
    <listitem><para>A package can also contain more packages in a subfolder. For example,
      <classname>java.awt.event</classname> contains classes for events, within the awt package.
      </para></listitem>
    <listitem><para>Use the <literal>import</literal> keyword to load in packages.</para></listitem>
    <listitem><para>Multiple classes can be loaded using the <literal>*</literal> character. So,
      for example,<literal>import java.awt.*;</literal> imports all the classes in the awt package,
      but please note, it does not include classes in the sub-packages, so, you must explicitly
      <literal>import java.awt.event.*;</literal>.</para></listitem>
    <listitem><para>In a source file, if no package name is defined on an import e.g.
      <literal>import SomeClass;</literal> then it is assumed that the class
      <classname>SomeClass</classname> is in the same directory as the source file.</para></listitem>
</itemizedlist>
</para>
</sect2>
```

(a)



(b) (c)

Figure 1. The DocBook conversion from (a) XML single-source to (b) HTML (c) PDF documents.

Figure 1(a) shows the XML format of DocBook, where the content is described, without the use of any style information. XML tags such as `<para>,<sect1>,<title> etc.` describe the structure of the document, and XML tags like `<classname>,<literal> etc.` describe concepts directly related to the

content itself. Once the external styles are applied, the Figure 1(b) HTML output and Figure 1(c) PDF output result.

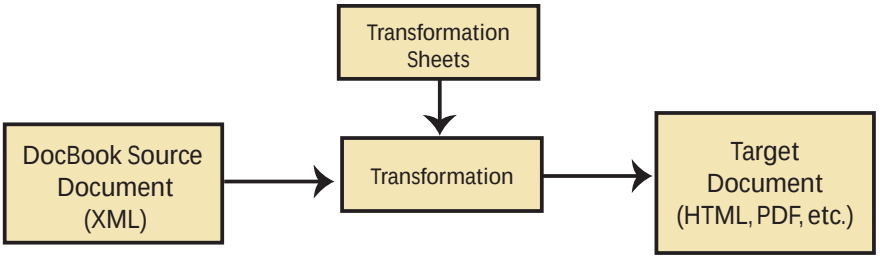


Figure 2. An overview of the XML conversion to HTML/PDF.

In this implementation Saxon [3] is used to validate the XML Document against the DocBook DTD (Document Type Definition) and to apply the DocBook stylesheets. At the validation stage, the source XML document is checked for valid syntax. Open and closing XML tags are strictly checked to insure that the document structure is correct. The conversion to the target format will not be performed if the document is not correctly formatted. DocBook provides several Transformation Sheets for converting DocBook XML to the Target Document. Some of these transformations are FO (for PDF conversion) and HTML (for html conversion), and within HTML there can be chunked documents (split into sections), or chunking with profiling (user-based presentation). The Transformation is performed by a transformation engine, such as Saxon [3] or Xalan [4]. The HTML document can be combined with Cascading Style Sheets (CSS) to provide advanced styling of the final HTML document.

3 Advanced Conversion Facilities

Profiling

DocBook allows profiling of the source document through user-profiles. For this module, user profiling has been used to generate an introductory and advanced version of the module notes. Each advanced paragraph is tagged using: `<para userlevel="advanced"> text </para>`. When the target document is generated for a "beginner" user-level, then the advanced topics are omitted. This is a very useful feature in this module, as it allows two parallel versions of the document, created from the one source, allowing students to examine the notes at an introductory level, and later return to deal with the advanced topics.

Interactive Content

The DocBook XML format allows specific actions to occur when converting a single XML document source to different formats using the `<?dbhtml command1 ?><?dbfo command2 ?>` tags, where command1 is called on the conversion to HTML and command2 is called on the conversion to PDF. This facility allows interactive content to be embedded into the HTML format, while a suitable replacement can be inserted for the document PDF version. For this module, on-line XML self-assessments forms are inserted into the HTML, allowing the student to fill in an assessment on-line and have it corrected, while a non-interactive version is inserted into the PDF document, allowing the student to complete the interactive version at a later stage.

Accessible Content

Since the content is almost completely separated from the style, this allows any style to be applied to the document with a minimum of effort. This has significant advantages for disabled users. The DocBook conversion tools automatically provide alternative image information, but tag information is also carried right through to the HTML document, in the HTML style class information. This means that the content description is retained, even through the conversion. The academic can provide a personalised high-

contrast CSS for inclusion, or even a personalised XSL for conversion. In the future VoxML could be used.

Customisation

The general DocBook format acts as a template to which advanced features can be added. For example, the `<classname>Button</classname>` tag, used as standard with DocBook can be extended to the format `<classname package="/java/awt">Button</classname>` to allow automatic hyper-linking of the Java SUN API documentation directly. This is achieved by editing the XSLT transformation document:

```
<xsl:template match="classname">
  <xsl:choose>
    <xsl:when test="@package">
      <a><xsl:attribute name="href"http://java.sun.com/j2se/1.4.1/docs/api/
        <xsl:value-of select="@package"/>.html
        </xsl:attribute><xsl:attribute name="target">_APIDOCS</xsl:attribute>
       <img src="images/derek_class.gif" border="0"/>
       <xsl:call-template name="inline.monoseq"/>
     </a>
    </xsl:when>
    <xsl:otherwise>
      <img src="images/derek_class.gif" border="0"/>
      <xsl:call-template name="inline.monoseq"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

The XSL Transformation rules allow very advanced conversion rules to be developed that have relevance to the content of the module.

4 Content Protection

Since the course module was developed to be presented remotely, it was necessary to embed some form of content protection. This has been achieved in two ways: (1) For the PDF version of the notes, a server-side processing engine was built using Java Servlets and the iText (Free Java-PDF Library)[6] API. When the complete module PDF format document is generated using DocBook it can take 3-4 minutes for the conversion to take place. It is not acceptable for the user to wait while a personalised version is created. Since there are 110+ students in the module, it is not possible to generate a separate version for each student. A system has been developed, as illustrated in Figure 3.

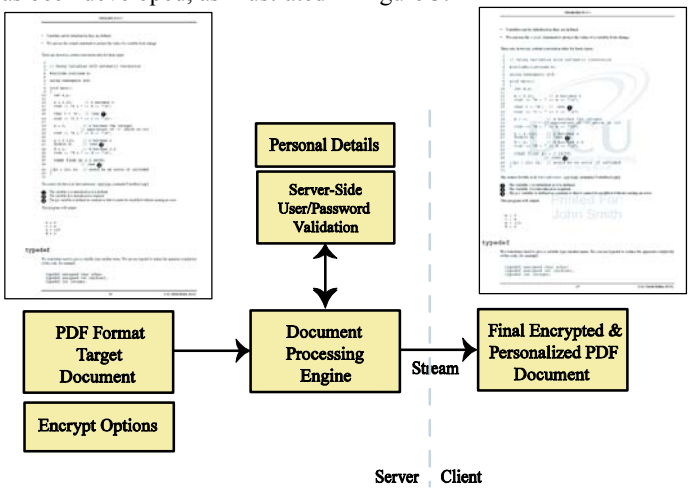


Figure 3. The real-time server-side PDF personalisation and encryption, showing the watermark added to the client version of the PDF document.

To combat this problem a single PDF document is generated off-line and is personalised at download time (taking 3-5 seconds) to add a personalised watermark, to encrypt the document using 128-bit encryption and to add advanced file properties (Allow printing, allow modify contents, allow copy, allow modify

annotations, allow fill forms, allow screen readers, allow assembly and allow degraded printing). Figure 3, illustrates the steps that take place in the personalisation of the document. The original PDF document and encryption options are passed to the Document Process Engine (DPE) that was developed using the iText API. The DPE forces the user to login to the teaching environment, where their personal details are recovered and embedded into the document as a watermark, a licensing cover page and encryption username/password. The document encryption prevents this information from being removed.

The HTML version of the document has also been protected as much as is possible, with the HTML documents maintained behind a protected section of the web-site, where the user must login to access the notes. Once the user has logged in, their personal information is inserted seamlessly into the document using Java servlets at download time, through HTML image tag execution and author information on each page. The HTML document is chunked into many sections, making download tedious. All links in the document are non-relative, i.e. they are tied to the course web-site. Typically this would be hard to maintain, but since the notes are generated dynamically it is much easier.

5 Discussion

A system has been presented for single-source publishing using the XML DocBook standard. It has proven stable and successful since implementation for this module. While the advantages of this approach have been presented so far in this paper, there are also several difficulties:

- There are few suitable WYSIWYG editors. The problem is that it is not possible to make assumptions about the documents that are to be edited by the user, or what the user is trying to achieve with the XML content. [8]
- Since the conversion rules are so flexible there is not one single conversion mechanism. Setting up the conversion tools is also quite complex. The set-up time for the technologies can be significant.
- The DocBook standard is particularly applicable to computing/engineering content. It would take some effort to establish tags for other content types (chemistry etc.)

Once a system is in place, modifications to the content can be easily applied and batch files can be written to automate deployment of generated content. There is a significant overhead in adding the advanced features listed here, such as user management, content protection and customisation. Once these features are in place further advanced features can be developed.

References

1. OASIS DocBook, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=docbook
2. LaTeX, http://www.latex.org/
3. Saxon, http://saxon.sourceforge.net/
4. Xalan, http://xml.apache.org/
5. Bruno Lowagie, iText, a Free Java-PDF Library, http://www.lowagie.com/
6. Apache XML Project, FOP, http://www.apache.org/fop/
7. Michael Kay's Saxon, http://saxon.sourceforge.net/
8. Villard, L. and Layaida, N. An Incremental XSLT Transformation Processor for XML Document Manipulation. WWW 20002, May 7-11, 2002, Honolulu, USA.