XML-Based Offline Website Generation

Florence Armardeilh

Information Technology and Electrical Engineering
The University of Queensland
St. Lucia QLD 4072, Australia

florence@dstc.edu.au

Peter Becker

Distributed Systems Technology Centre
The University of Queensland
St. Lucia QLD 4072, Australia

pbecker@dstc.edu.au

Abstract

The approach and the tool XWeb[1]*, presented in this paper, shows one way to create websites from XML and other input files which can then be uploaded onto standard web-servers. The system uses an extra input file describing the structure of the content and the processes for creating the website. This information is also used to create the navigational elements in the output. Generating the content offline avoids having additional requirements on the server side such as CGI interfaces or Servlet engines.*

Keywords Document Management, XML, Hypermedia, Website Generation, Processing Model

1 Introduction

HTML was originally designed for writing documents with simple structures that are hyperlinked with other documents. This design was very successful and created what we know nowadays as the World Wide Web. Unfortunately HTML did not address two important issues when managing documents on a larger scale: there is no support for navigational structures outside the documents and content and layout information is highly intermixed.

The screenshot in Fig. 1 shows a typical webpage, which contains navigational structure on the left hand side, using a nested layout for two levels of navigation, a number of layout elements plus information content. The information content in this example comes from two different sources: a description of the person and a BibTeXML[2] file describing the publications of the whole workgroup, a single publication can appear on multiple pages if it has more than one author.

Separating these different aspects of a site has been addressed in a number of ways, the most common approaches being the use of frames as an HTML extension and the use of scripting languages to create websites dynamically from databases or files on the server. Frames introduce a number of technical problems on the client side and often introduce accessibility issues for the visually impaired. They also do not provide a linkage between the document structure given and the navigation structure implemented as an HTML document in one of the frames. It is left to the website creator to ensure consistency between the two.

Using scripting languages like PHP, ASP, JSP, Perl, Python or others on the server side introduces two issues. The first is that the server needs to be capable of running the scripting language used via some interface, usually the Common Gateway Interface (CGI). This raises the requirements on the server side, especially if the data itself is stored in a RDBMS instead of plain files – quite common with this approach. Another problem is the computational effort. Creating pages dynamically means running a script every time a page is requested, which increases the need for computational power on the server side significantly.

The most common way to separate content and layout in the more modern HTML versions is the use of Cascading Style Sheets (CSS). CSS introduces layouting information that can be used to change the rendering of different HTML elements, either by tag names, by their position in the document or by additional information attached to them. CSS is a quite powerful way to define how elements are to be rendered in a browser, although it does not allow structural changes.

[1] `http://xweb.sourceforge.net`
[2] `http://bibtexml.sourceforge.net`

Proceedings of the 7th Australasian Document Computing Symposium,
Sydney, Australia, December 16, 2002.

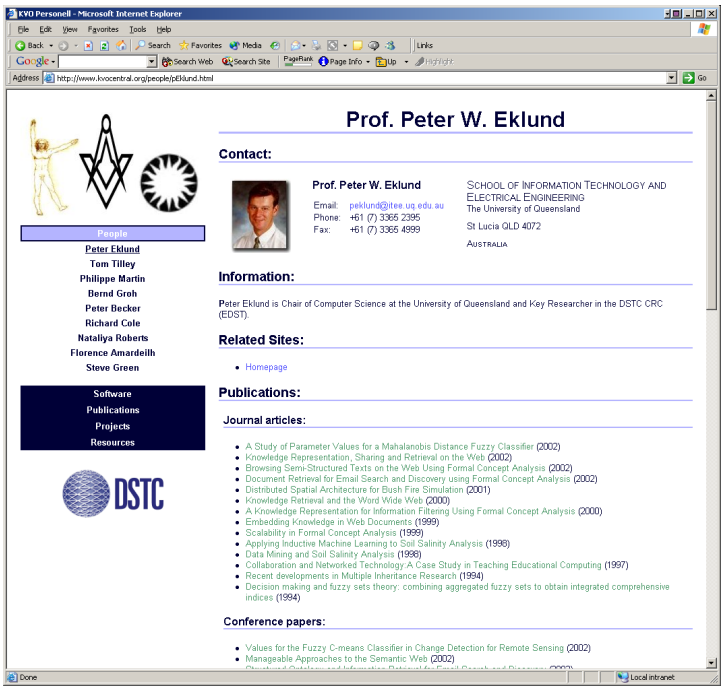

Figure 1: A page of a website presenting an academic working group.

2 XML

The World Wide Web Consortium (W3C)[3] proposes XML as an approach to separate content and layout. XML itself offers a simple, tree-based data model called the Document Object Model (DOM), which is mapped into an Unicode-based text format.

The eXtensible Stylesheet Language for Transformations (XSLT) can be used to transform XML documents into any other format, including standard HTML files and other web-based formats. The XML input can have a structure that reflects the content semantics, while all layout information is moved into XSLT programs. These XSLT programs can then be applied either on the client side (i.e. in a browser) or on the server side.

3 XWeb

The approach and the tool presented here try to fulfil the following goals, addressing some of the issues discussed above:

- XML and XSLT shall be leveraged to separate content and layout;
- CSS shall be used to define the rendering details;
- the website navigation shall be generated from the content structure;
- the processing should be done offline, i.e. a result shall be created using a compiler-like approach, which can then be uploaded onto a standard webserver.

The software architecture implemented uses a three-layer model to fulfil these design goals in a flexible way. The lowest layer is a Java-based processing engine, which collects all information needed for the creation of the navigation and calls different processors to do the document processing, the most important processor being an XSLT-processor. The second layer consists of XSLT programs, XML formats and CSS files to create websites with specific structures and designs. The XSLT programs can use parameters to allow flexible layouts, e.g. for deciding the position of the navigation on the pages. As a third layer, specific graphical frontends for creating those sites can be implemented.

This layered approach allows for a high level of abstraction and therefore reuse in the lower layers and also accomodates different user types with different needs for control and comfort.

The approach taken for defining a website in XWeb is using a separate file defining the structure of the content in an XML-based format. The toplevel element of the file is called `website` and has some attributes defining input and output locations as well as the target webspace. Below this, the two elements `structure` and `layout` define the content structure and the layout resp., the latter can be given multiple times to create different versions of the same site, e.g. for different browsers.

[3] `http://www.w3c.org`

3.1 The Content Structure

The two core concepts in the structure definition are denoted by `section` and `entry` elements. A section defines a part of the website, which usually has a name attached to it used in the navigation rendering, and usually maps into subdirectories in the input and output. This information is attached as attributes on the `section` element. A `directory` is similar to a section, but will not create a navigation element.

Entries define single pages in the website. They are also named for the rendering of the navigation, they need an input and an output filename and an identifier for the type of process that has to be executed to transform the input file into the output file. This process has to be defined in the layout section later on. If a single file should be omitted in the navigation, the `file` element can be used instead.

The code in Fig. 2 defines a section of the website with three pages and two files which will not be shown in the navigation, in this case they are actually linked from the output file created by the entry above. Two different processes are used, attached to the files via their type definitions, the entries are all of type "XHTML" while the other two files are of type "copy". The processes for these file types are then defined in the layout sections.

Compared to using directory structures for defining the website navigation and file extensions for the processes, this XML structure gives more control on the way files are processed and offers ways to separate the input and output structure, as well as the input and output file names and the names rendered in the navigation structures. Additional information can be added in the same way, in its current state XWeb allows adding types to the section (for rendering graphical buttons) and an additional *title* attribute can be given to distinguish between a long and a short name for an entry, usually used with banners across the pages.

3.2 Layout information

The `layout` sections of the XML website file define how the website should be generated from the given content. In the most simple case one `layout` section defines a process for each of the file types used in the `structure` part, as shown in Fig. 3.

The layout definition given here tells XWeb that all files declared as having the type "copy" should just be copied from the input file name and location to the output file name and location. The files of type "XHTML" are supposed to be processed by an XSL process, using a specific stylesheet given as attribute of the `xsl` element. The second attribute *navigationElement* tells XWeb to insert the navigation information created from the structural information into the input document before calling the XSLT processor. This process will be discussed in the next section.

There are other options for processing documents and creating additional layout elements. For brevity they will be only listed here, please refer to the XWeb project site for details.

- *XSLT chains:* Multiple XSLT stylesheets can be concatenated to separate different aspects of a transformation and thereby increase reuse.
- *Other process types:* SVG images can be rendered into bitmap formats and external programs can be called.
- *Rendering buttons and banners:* Images can be rendered from SVG content or using a simple XML format. Replacing text with names from the navigation allows creating buttons and banners.

4 Website Generation Process

After the input file has been parsed into a DOM document, the layout definitions are used to create processing objects and its structural part is traversed twice. During the first traversal, XWeb collects all information necessary for creating the navigation and adds it into the DOM. Most importantly this is the URL of the target location for each file. In addition to this, images for buttons and banners will be created and the information about their location added into the DOM.

During the second traversal the documents are processed. For all `entry` and `file` elements the according processes are called. They can use the information in the DOM, including the parts calculated or generated during the first traversal.

The most interesting sub-process is the `xsl` process. The XSLT processing can be done by calling a single stylesheet on the input file, with the output written to the output file given in the structural part, but there are two further options: XSLT chains and the addition of the navigation information. If multiple `xsl` tags are nested, XWeb will parse the file, pass it as input into the XSLT processor using the innermost stylesheet and then pass the output stream into an XSLT processor again, this time with the next outer stylesheet being used. The result of the outermost XSLT process is then written into the output file. In this way multiple transformations are applied without the need for temporary files, using the Simple API for XML (SAX) as streaming model.

If one of these transformation steps needs access to the navigation information collected by XWeb before, the relevant section of the website DOM can be added into the XML stream. Whenever the attribute *navigationElement* is defined on the `xsl` element, XWeb will scan the input for this XSLT

```
<section name="Docu" sourceDir="documentation" targetDir="doc">
    <entry name="DocMain" sourceFile="index.xhtml" targetFile="index.html" type="XHTML"/>
    <entry name="Installation" sourceFile="install.xhtml" targetFile="install.html" type="XHTML"/>
    <entry name="Tutorial" sourceFile="tutorial.xhtml" targetFile="tutorial.html" type="XHTML"/>
    <file sourceFile="xweb_tutorial.zip" targetFile="xweb_tutorial.zip" type="copy"/>
    <file sourceFile="xweb_tutorial_output.zip" targetFile="xweb_tutorial_output.zip" type="copy"/>
</section>
```

Figure 2: A definition of a website section.

```
<layout>
    <documentStyle type="copy">
        <copy/>
    </documentStyle>
    <documentStyle type="XHTML">
        <xsl stylesheet="stylesheets/orangeLayout.xsl" navigationElement="html"/>
    </documentStyle>
</layout>
```

Figure 3: A definition of a website layout.

transformation for the first occurance of an element with the given name. If this is found, a copy of the `structure` part, including the collected additional information will be added as a child into the XML stream. In this way the XSLT stylesheets can get access to the information they need to create the navigation. How exactly this information is used, and how the look and feel of the website will be, is left up to the stylesheet author. In combination with the buttons and banners created by the SVG image renderer this approach is very flexible.

5 Further Research and Development

The processing model used in XWeb is in many ways limited. The streaming approach taken for the XSLT processing does not extend into other areas, for a number of purposes one would like an extended processing model, based on XML and binary streams. This should include forking (e.g. splitting XHTML with SVG and MathML islands into separate streams), merging (combining the information from different content sources, or adding the navigation information) and multiplexing (multiple streams of the same type at once, where the number depends on the input, not the process setup).

Some Open Source projects and research groups have a stream based processing model, e.g. Lagoon[4][3] and Transmorpher[5][1]. Their models are more advanced than XWebs document-centric model, but they are not sophisticated enough to support the whole range of XML-based processing and the integration of the POSIX stream model. The processing model for this will be complex, but it should map into a reasonably easy programming system for processing modules, implemented by mapping named output streams of one process onto named input streams of one or more others.

Another quite promising approach grounds web sites on a knowledge base structure using RDF[6] as done in the Haystack project[2]. RDF offers the ability to interlink elements with a large number of different relations, which then can be used to create specific pages. This gives the option to create user-centric portal sites, although the classic tree-based navigation approach gives a more common structure and might therefore be easier to use.

6 Acknowledgements

XWeb itself is not much more than some glue between a set of Open Source libraries. Thanks to all people who worked on the main libraries used: Saxon (XSLT processing), Xerces (XML parsing), JDOM (Java-based DOM access) and Batik (SVG rendering). Thanks also to all people on similar projects who spend time discussing different approaches with the authors and all users giving feedback.

References

[1] Jérôme Euzenat and Laurent Tardif. XML transformation flow processing. In *Proceedings of the Extreme Markup Languages*, 2001.

[2] David Huynh, David Karger and Dennis Quan. Haystack: A platform for creating, organizing and visualizing information using rdf. Presented at the Semantic Web Workshop 2002.

[3] Mikael Ståldal. Presenting XML documents on different media with stylesheets. Technical report, KTH, Stockholm, Sweden, 2000.

[4] `http://lagoon.sourceforge.net`

[5] `http://transmorpher.inrialpes.fr/`

[6] `http://www.w3.org/`